PROGRAMMAZIONE WEB

LEZ1 – LEZ2 – LEZ3 (DAL 5/03/2024 AL 12/02/2024)

URL:

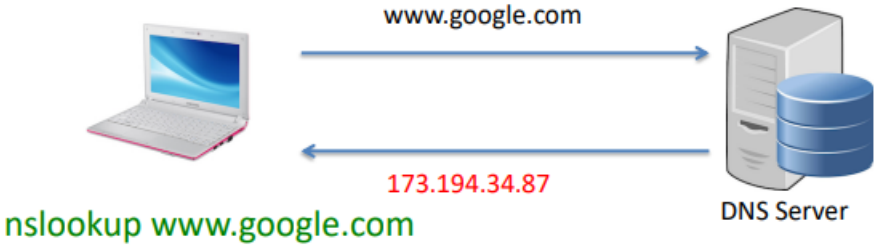
https://internet.uniroma2.it/esame/programmazione-web/

https://ing.uniroma2.it/area-studenti/guida-dello-studente/

WWW:

Il World Wide Web (WWW) è un insieme di pagine web ed altri documenti collegati tra loro tramite collegamenti ipertestuali(link). Il protocollo principale per scambiare documenti è HTTP – HyperText Transfer Protocol.

DOMAIN NAME SERVER:
Dare un nome ai server per l'uomo, ai computer serve un indirizzo numerico per comunicare con le altre macchine. Esistono dei computer che traducono nomi in indirizzi IP.



SERVER WEB:
Sono computer che rispondono a delle richieste di documenti. In realtà chi offre il servizio è un programma: il server web.
Aspetta una richiesta, cerca un documento e lo invia al richiedente. Il server web deve parlare HTTP, spesso lo chiamiamo Server HTTP. I computer server non sono particolari, ogni computer può diventare un server.
Server(programmi) famosi sono apache e nginx.

CLIENT WEB:
I software che fanno richieste HTTP ai server si chiamano client

- sono clienti dei server
- I più comuni client web sono i Browser.

Richiesta di un documento:

- L'utente inserisce un indirizzo
- Il browser chiede un documento al server
- Il server glielo spedisce
- Il browser lo mostra all'utente

BROWSER: I browser parlano HTTP come i server web. Sia le richieste che le risposte sono gestite mediante HTTP.

IN UNA PAGINA WEB: HTML per il contenuto della pagina, CSS per lo stile e Javascript per aggiungere interattività.

HTML

Sigla per HyperText Markup Language.
E' uno standard del W3C, ed è lo standard più utilizzato per la creazione di pagine web visualizzabili attraverso i browser.

Il markup definisce elementi mediante tag.
Gli elementi possono essere annidati.
Possibilità di fornire informazioni addizionali su un elemento definendo degli attributi.
Gli elementi formano un albero.

Struttura documenti html:

- L'elemento <html> è la radice dell'albero e contiene tutti gli altri
- L'elemento <head> fornisce informazioni circa il documento, contiene metadati relativi alla pagina(descrizione, charset, autore, ecc), contiene il titolo:

```
<head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="style.css">
    <title>La mia prima pagina</title>
</head>
```

- L'elemento <body> contiene il contenuto visibile, normalmente è diviso in sezioni:

```
<body>
  <header>
    <!-- Intestazione del corpo della pagina web
        (ad es. logo, barra di navigazione) -->
  </header>
  <main>
    <!-- Sezione principale del corpo della pagina web
        (dove si trova la maggior parte del contenuto) -->
  </main>
  <footer>
    <!-- Piè di pagina del corpo della pagina web
        (ad es. informazioni sul copyright) -->
  </footer>
</body>
```

La dichiarazione <!DOCTYPE> serve a dire al browser il tipo di documento HTML5. La dichiarazione non è case sensitive.

➔ Per organizzare il contenuto i tag principali di HTML5 sono:

- <main>
  - Specifica il contenuto principale di un documento, ci può essere un solo elemento in un documento
  - NON deve essere un discendente di un elemento <article>, <aside>, <footer>, <header>, <nav>
- <aside>
  - Elementi "marginali" rispetto al contenuto
- <nav>
  - Sezione che permette la navigazione nel sito
  - Tipicamente al suo interno si mettono delle liste
- <header>
  - Due usi comuni sono header della pagina e header di una sezione
  - Se usato in un articolo tipicamente contiene titolo, autore, data, etc.

- <footer>
  - Usi comuni sono fondo pagina e fine articolo
  - Contiene ad esempio autore, copyright, documenti correlati
- <section>
  - Sezioni tematiche: possono contenere heading, paragrafi e in genere tutti gli elementi che abbia senso raggruppare
- <article>
  - Contesti operativi: blog, giornali
- <figure>
  - Un'immagine e una didascalia possono essere raggruppate in un elemento

➔ Altri tag:

<p> represents a paragraph.
<hr> represents a paragraph-level thematic break
<pre> represents a block of preformatted tex
<blockquote> represents content that is quoted from another source
<ol> represents a list of items, where the items have been intentionally ordered
<ul> represents a list of items, where the order of the items is not important
<li> represents a list item
<dl> represents an association list consisting of zero or more name-value groups
<dt> represents the term, or name, part of a term-description group in a description list
<dd> represents the description, definition, or value, part of a term-description group in a description list
<figure> represents some flow content
<figcaption> represents a caption or legend
** ** no special meaning at all
<main> represents the main content of the body of a document or application
<a> Iperlink
<abbr> abbreviazione
<b> evidenziare parole, keyword
<bdi> Indicates text that may have directional requirements
<bdo> Bidirectional override; explicitly indicates text direction (left to right, ltr, or right to left, rtl)
**
** Line break
<cite> Citation; a reference to the title of a work, such as a book title
<code> Computer code sample
<data> Machine-readable equivalent dates, time, weights, and other measurable values
<del> Deleted text; indicates an edit made to a document
<dfn> The defining instance or first occurrence of a term
**** Emphasized text
<i> Alternative voice (italic)
<ins> Inserted text; indicates an insertion in a document

<kbd> Keyboard; text entered by a user (for technical documents)
<mark> Contextually relevant text
<q> Short, inline quotation
<ruby>, <rt>, <rp> Provides annotations or pronunciation guides under East Asian typography and ideographs
<s> Incorrect text (strike-through)
<samp> Sample output from programs
<small > Small print, such as a copyright or legal notice (displayed in a smaller type size)
**** Generic phrase content
**** Content of strong importance
<sub > Subscript
<sup> Superscript
<time> Machine-readable time data
<u> Underlined
<var> A variable or program argument (for technical documents)
<wbr> Word break

➔ Commenti: <!-- commento -->

➔ Caratteri Unicode: alcuni caratteri devono essere sostituiti.
La sequenza di escape inizia con & e si chiude con un ;
es: © &#169

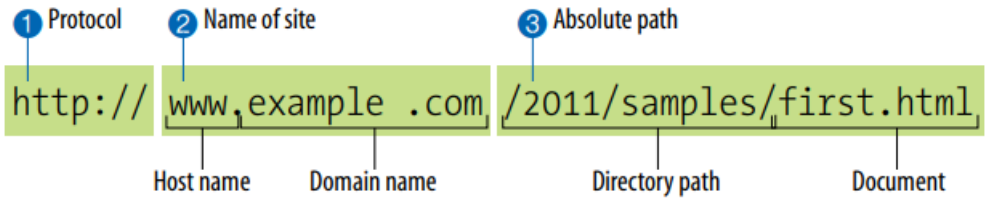
➔ Altri tag base sono:
- ➢ <img src=”nome.gif” alt=”descrizioneImmagine”>
  - Utilizza i due attributi src (per la url dell’immagine) e alt (per il testo alternativo in caso non fosse possibile visualizzarla)

- ➢ ….
  - Un esempio è <a href=”https://www.w3schools.com”>visita w3schools</ah>
  - L’attributo href specifica la url di un documento nel web
  - Due modi per specificare le url:
    URL assolute, che iniziano con http://
    URL relative, per esempio alla URL del documento corrente

➔ <table> in HTML:
- Ogni riga è definita tramite il tag <tr>.
- L’intestazione della tabella è definita con il tag <th>. Di default, l’intestazione viene mostrata centrata e in grassetto.
- Una casella della tabella viene definita con il tag <td>.

INDIRIZZI DELLE PAGINE WEB: URL

Ogni pagina web e ogni risorsa ha un indirizzo, questo si chiama URL (Uniform Resource Locator).



File di default: alcune URL non includono il nome del file (indicano solo una directory). Se una richiesta non presenta esplicitamente il file il server web cerca il file index.html

Link esterni: un esempio è http.//ppl.eln.uniroma2.it ; devo usare URL assolute in quanto si trova su un server web diverso.
Link interni: link a una risorsa nello stesso server web (spesso nello stesso sito). Nel caso di link interni posso omettere il protocollo ed il nome del server, basta indicare il path del file.

GIT

Git è un controllo di versione open source e distribuito che aiuta a:

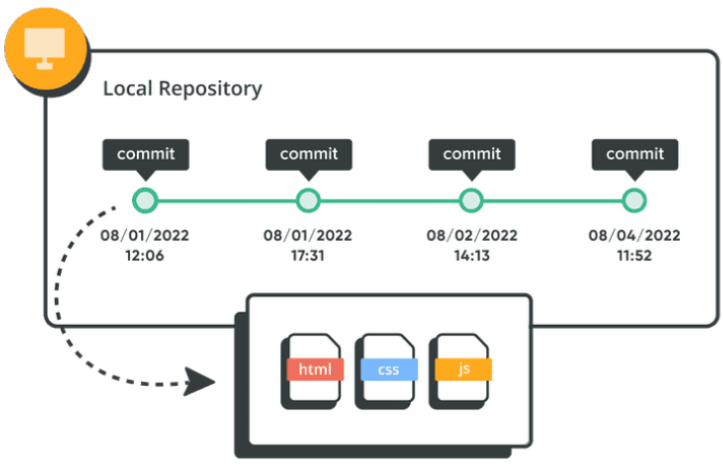
- Tenere traccia della cronologia del codice
- Collaborare al codice come team
- Scoprire chi ha apportato quali modifiche
- Distribuire il codice nella gestione temporanea o nella produzione

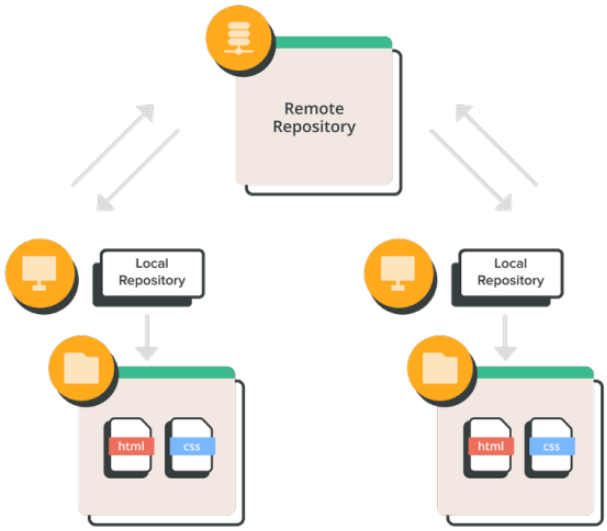
Le sue componenti sono:

- Repository, il "contenitore" che tiene traccia delle modifiche
- Working tree o directory, costituito da file su cui si sta lavorando
- Index o Staging Area, dove vengono preparati i commit

Classico flusso di lavoro:

1. Modificare i file nel working tree
2. Eseguire lo staging delle modifiche che si desidera includere nel commit successivo
3. Eseguire il commit delle modifiche, il commit prenderà i file dall'index e li memorizzerà come snapshot nel repository



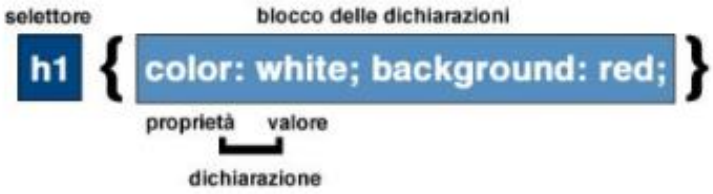


CSS

Sta per Cascading Style Sheets, ed è un linguaggio indipendente.

CSS è formato da:

- Il selettore: ovvero l'elemento/ gli elementi da modificare;
- Il blocco delle dichiarazioni: delineato da parentesi graffe {} dentro il quale l'utente indicherà come vuole cambiare l'elemento selezionato, tramite:



la proprietà: indica cosa l'utente vuole cambiare di quell'elemento;
il valore: della proprietà impostata come l'utente vuole cambiarla.

Funzionamento: Scrivere un documento HTML 2, scrivere le regole CSS, "Agganciare" le regole all'HTML.
Stili nell'head tramite external style sheet tramite

<link rel="stylesheet" type="text/css" href="mystyle.css">

(l'elemento <link> allega un file alla pagina corrente e va posizionato nella sezione head)
Elemento link usato anche per le favicon tramite

<link rel="icon" href="demo_icon.gif" type="image/gif">

Stile inline: un esempio è <h1 style="color:red;">titolo</h1>
Non andrebbe mai usato, si usa in casi particolari come l'override mirati di regole esterne.

➔ SELETTORI SEMPLICI

Il selettore indica a quale elemento deve essere applicata la regola (lo stile). È possibile selezionare gli elementi in tre modi differenti:

1) Elemento , seleziono tutti gli elementi di quel tipo

Esempio: seleziono tutti i p

```
p {
    color: red;
    text-align: center;
}
```

```
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

2) Classe , seleziono tutti gli elementi di una classe
   - Li individuo con l'attributo class ed un identificatore
   - Uso come selettore l'identificatore preceduto dal punto

Esempio

```
<h1 class="center">Heading</h1>
<p class="center">A paragraph</p>
```

```
.center {
  text-align: center;
  color: red;
}
```

```
p.center {
  text-align: center;
  color: green;
}
```

3) Id , seleziono l'elemento con quell'id scrivendo l'id preceduto da #

Esempio: ...

```
#para1 {
    text-align: center;
    color: red;
}
```

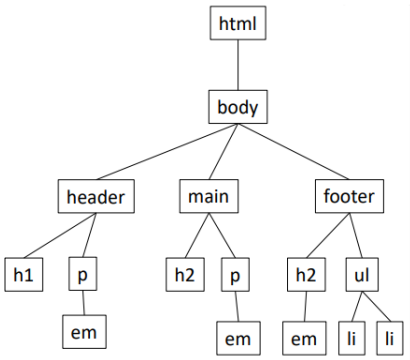
RAGGRUPPARE SELETTORI: Si possono anteporre più selettori ad un blocco di dichiarazioni, i selettori si radunano con la virgola

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

➔ SELETTORI E DOM

Relazioni nel DOM:

1. Discendenti, gli elementi contenuti in un elemento sono i suoi discendenti.
2. Figli, discendenti diretti e viceversa si dice genitore(parent)
3. Antenati, gli elementi sopra nell'albero
4. Genitori, elementi direttamente sopra
5. Fratelli, elementi con lo stesso genitore



SELETTORI COMPOSTI:

1) SELETTORI PER DISCENDENTI (tramite spazio)
- Sintassi: A B
- Seleziona tutti gli elementi B che sono discendenti di A, indipendentemente dal livello di profondità nella gerarchia.
- Esempio: p em {color: red;} Questo selettore applicherà lo stile a tutti gli elementi che si trovano all'interno di un elemento , anche se non sono figli diretti.

2) SELETTORI PER FIGLI (tramite >)
- Sintassi: A > B
- Seleziona tutti gli elementi B che sono figli diretti di A, escludendo quelli che sono più profondi nella gerarchia (come nipoti o pronipoti).
- Esempio: p > em {color: red;} Questo selettore applicherà lo stile solo agli elementi che sono figli diretti di , ignorando quelli più profondi nella gerarchia.

3) SELETTORI PER FRATELLO IMMEDIATAMENTE SUCCESSIVO (tramite +)
- Sintassi: A + B
- Seleziona l'elemento B che è il fratello immediatamente successivo di A, cioè l'elemento che compare subito dopo A nello stesso livello gerarchico.
- Esempio: div + p {color: red;} Questo selettore applicherà lo stile solo al primo che segue immediatamente un .

4) SELETTORI PER TUTTI I FRATELLI SUCCESSIVI (tramite ~)
- Sintassi: A ~ B
- Seleziona tutti gli elementi B che sono fratelli successivi di A, quindi non solo il primo fratello immediato, ma anche quelli successivi nello stesso livello gerarchico.
- Esempio: div ~ p {color: red;} Questo selettore applicherà lo stile a tutti gli elementi che si trovano dopo un nello stesso livello, indipendentemente dal fatto che siano immediatamente successivi o meno.

LEZ4 – 14/03/2024

PSEUDO CLASSI

Una pseudo-classe viene utilizzata per definire uno stato speciale di un elemento (mouse over , visited e unvisited link , focus)

```
selector:pseudo-class {
  property:value;
}
```

```
/* mouse over link */
a:hover {
  color: #FF00FF;
}
```

```
/* visited link */
a:visited {
  color: #00FF00;
}
```

PSEUDO ELEMENTI

Uno pseudo-elemento viene utilizzato per dare uno stile alle parti specifiche di un elemento:

- Disegna la prima lettera o riga di un elemento
- Inserire un contenuto prima o dopo un elemento.

Selettori:

1) ::after
2) ::before
3) ::first-letter
4) ::first-line
5) ::selection

```
selector::pseudo-element {
  property:value;
}
```

```
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
```

SELETTORI CON ATTRIBUTI

È possibile impostare lo stile di elementi HTML con attributi o valori di attributo specifici.

```
[attribute] {
  property: value;
}
```

```
[attribute=value] {
  property: value;
}
```

```
img[alt] {
  background-color: grey;
}
```

```
a[target="_blank"] {
  color: red;
}
```

EREDITARIETA'

Alcune proprietà sono ereditate dai discendenti.

```
<!DOCTYPE html>
<html>
<head>
<style>

p{
  color:white;
  background:grey;
  border: medium solid black;
}

</style>
</head>
<body>
  <h1>This is a Heading</h1>
  <p>This is a paragraph <span>with a span</span> inside.</p>
</body>
</html>
```

CASCADE

Algoritmo che definisce come combinare valori di proprietà provenienti da fonti diverse

CONFLITTI

I conflitti di dichiarazione sono la normalità: possono applicare più stili allo stesso elemento, alcune proprietà le eredito.

Esempio:

```
p{
  color: blue;
  margin-left: 30px;
}

p{
  color: green;
}
```

```
p{
  color: blue;
  color: green;
}
```

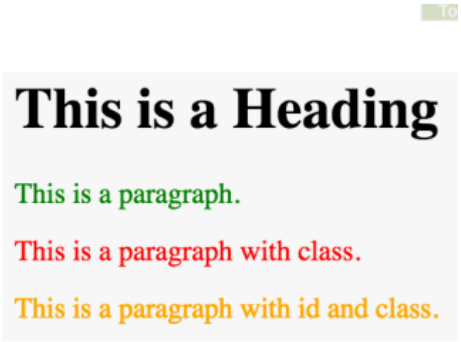
la seconda dichiarazione cancella la prima (domina quella più vicina all'elemento quindi il p sarà verde).
Attenzione: non è sempre così!

ESEMPI

CASCADE:

1)

```
<!DOCTYPE html>
<html>
<head>
<style>

#para3{
  color: orange;
}

.red{
  color: red;
}

p{
  color: green;
}

</style>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
<p class="red">This is a paragraph with class.</p>
<p id="para3" class="red">This is a paragraph with id and class.</p>

</body>
</html>
```



Non tutti i selettori sono uguali, alcuni hanno più "peso" di altri

2)

```
<!DOCTYPE html>
<html>
<head>
<style>
section{
  color: grey;
}

.news .mark{
  font-weight: bold;
}

.blog .mark{
  color: orange;
}

</style>
</head>
<body>

<h1>This is a Heading</h1>
<section class="news">
  <p>This is a paragraph in sec news.</p>
  <p class="mark">This is a another paragraph in sec news.</p>
</section>
<section class="blog">
  <p class="mark">This is a paragraph in sec blog.</p>
  <p>This is a another <span class="mark">paragraph</span> in sec blog.
</p>
</section>
</body>
</html>
```

!important

Se non voglio che una regola venga sovrascritta la posso dichiarare !important, questo si fa in casi particolare e di solito è sempre possibile evitarlo.
Esempio: p{ color: green !important; }

CALCOLO DELLA SPECIFICITA'

Ad ogni dichiarazione è attribuita una specificità misurata con quattro valori [a,b,c,d]

- ogni valore è indipendente dagli altri
- "a" è il valore più importante e "d" il meno importante
- si confrontano i valori più importanti e se uguali si passa a quelli successivi altrimenti termina

Calcolo dei valori:

a) 1 se la dichiarazione è inline, 0 altrimenti
b) Numero di selettori id
c) Numero di selettori di classe, attributo o pseudo-classe
d) Numero di selettori elemento o pseudo elemento

Esempio: style="" /* a=1 b=0 c=0 d=0 -> specificity = 1,0,0,0 */

UNITA' DI MISURA

Tra le unità di misura utilizzabili in HTML-CSS le più utilizzate sono: px (pixel) , em (unità di misura uguale al font size corrente) , % (relativa all'elemento genitore) , rem (root em, uguale alla dimensione em dell'elemento radice), vw(viewport width) e vh(viewport height) entrambe responsive perché forniscono un rapporto diretto con le dimensioni della Viewport (dove con viewport si intende identificare la dimensione della finestra del browser).

FORMATTARE IL TESTO

Proprietà relative ai font sono:

- font-family
- font-size
- font-weight
- font-style
- …

Proprietà relative al testo sono

- color
- line-height
- text-decoration
- text-align
- …

FONT-FAMILY

Forniamo una lista di "typeface" da applicare al testo come ad esempio:

```
body { font-family: Arial;}
code { font-family: Courier, monospace;}
p    { font-family: "Times New Roman", Times, serif;}
```

Le regole sono che tutti i font cominciano con lettera maiuscola tranne quelli generici, sono separati da virgola e se il loro nome contiene spazi vanno fra virgolette

FONT-SIZE

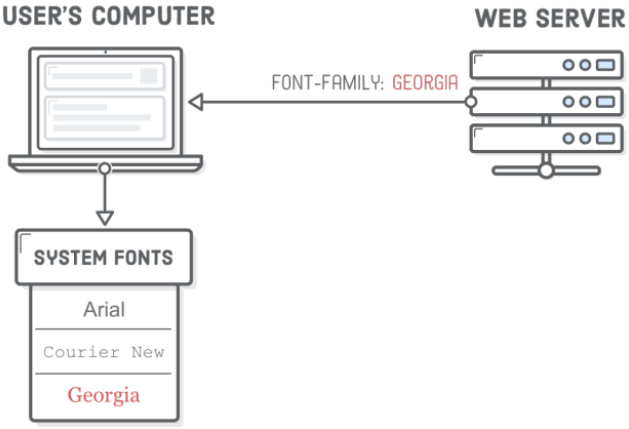
Specifica la dimensione del carattere. Ammette i valori seguenti:

- lenght unit | percentage
- xx-small | x-small | small | medium | large | x-large | xx-large,
- smaller | larger.

Alcuni esempi sono:

```
h1 { font-size: 150%; }
h1 { font-size: 22px; }
h1 { font-size: 1.5em; }
h1 { font-size: x-large; }
h1 { font-size: larger; }
```

FONT E SISTEMI OPERATIVI



Non tutti i font sono disponibili in quanto dipendono dal sistema operativo e sono installati nel pc.

Si possono prendere font da fonti web esterne per ovviare a questo problema.

TESTO E COLORI

- color specifica il colore del testo, viene ereditato dagli elementi e non influenza solo il testo. Esempio: h1{color:grey;}
- colori RGB: specifico il colore con 3 valori numerici (quantità di rosso,verde,blu sono 3 byte)
- colori RGBa: RGB + canale alfa per il controllo del livello di trasparenza (l'ultimo valore ha una scala che va da 0.0 a 1: con 0,0 oggetto completamente trasparente)

TEXT

- text-align utilizzato per allineamento orizzontale, i possibili valori sono left, right, center, justify
- text-decoration utilizzato per decorare il testo, i possibili valori sono none, underline, overline, line-through, blink
- text-transform utilizzato per cambiare maiuscole e minuscole, i possibili valori sono none, capitalize, lowercase, uppercase
- text-indent utilizzato per l'identazione del primo rigo, i possibili valori sono lenght measurement, percentage
- text-shadow utilizzato per ombra sul testo, i possibili valori sono 'horizontal offset', 'vertical offset', 'blur radius' , 'color'
- letter-spacing e word-spacing utilizzati per cambiare spazio tra lettere e parole, i possibili valori sono lenght measurement, normal.
  Esempio:
  ```
  p { letter-spacing: 8px; }
  p { word-spacing: 1.5em; }
  ```

RIASSUNTO

font-family: font utilizzato, e.g. Helvetiva, Arial
font-size: dimensione del testo, e.g. 60px, 3em
font-weight: Quanto è grassetto il testo, e.g. bold
font-style: Che stile è il testo e.g. italic
line-height: spaziatura tra le righe, e.g. 2em

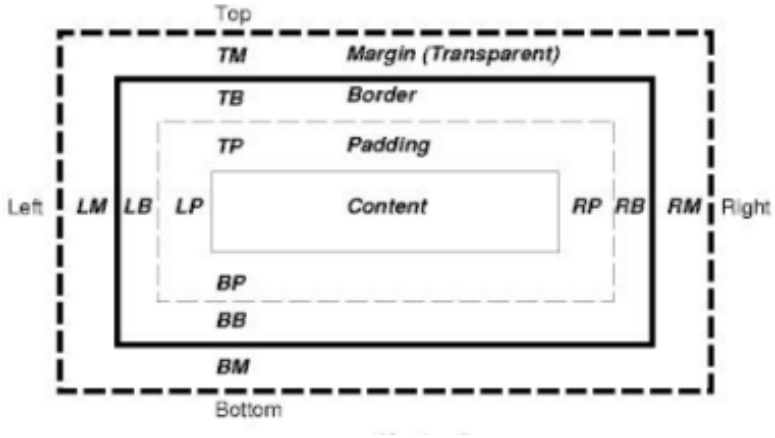
color: colore del testo, e.g. #000. #abedef

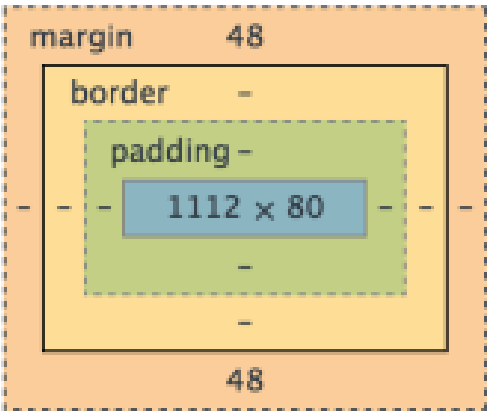
text-decoration: imposta effetti sul testo, e.g. underline, overline, none
text-align: Come è allineato il testo, e.g. center
letter-spacing: spaziatura tra le lettere, e.g. 5px
text-indent: rientro della prima riga, e.g. 2em
text-transform: trasforma il testo, e.g. upper-case, lowercase, capitalize
vertical-align: Allineare rispetto alla linea di base, e.g. text-top

LEZ5 – 19/03/2024

BOX MODEL

Un elemento HTML può essere considerato come una scatola.
Nella costruzione del design e del layout si parla quindi di box model.
La scatola è composta da margini, bordi, padding e contenuto.





DIMENSIONI

width

Values:	*length measurement* | *percentage* | auto | inherit
Default:	auto
Applies to:	*block-level elements and replaced inline elements (such as images)*
Inherits:	*no*

height

Values:	*length measurement* | *percentage* | auto | inherit
Default:	auto
Applies to:	*block-level elements and replaced inline elements (such as images)*
Inherits:	*no*

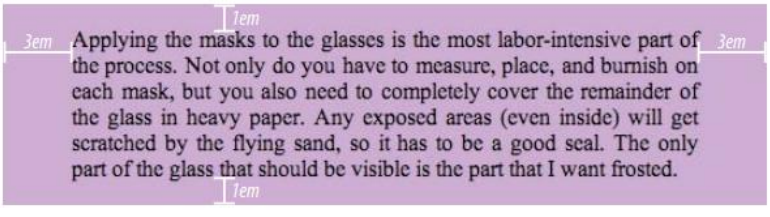
PADDING

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

Values:	*length measurement* | *percentage* | inherit
Default:	*0*
Applies to:	*all elements except table-row, table-row group, table-header-group, table-footer-group, table-column, and table-column-group*
Inherits:	*no*

padding

Values:	*length measurement* | *percentage* | inherit
Default:	0
Applies to:	*all elements*
Inherits:	*no*




```
blockquote {
  padding-top: 1em;
  padding-right: 3em;
  padding-bottom: 1em;
  padding-left: 3em;
  background-color: #D098D4;
}

blockquote {
  padding: 1em 3em 1em 3em;
  background-color: #D098D4;
}
```

MARGIN

margin-top, margin-right, margin-bottom, margin-left

Values: *length measurement* | *percentage* | auto | inherit

Default: auto

Applies to: *all elements*

Inherits: *no*

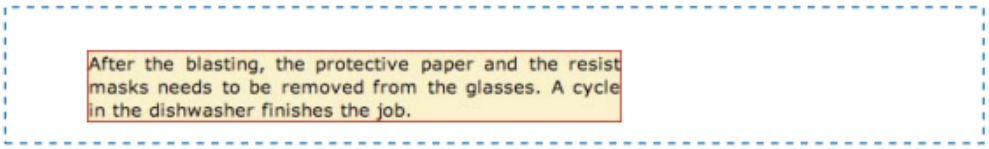
margin

Values: *length measurement* | *percentage* | auto | inherit

Default: auto

Applies to: *all elements except elements with table display types other than table-caption, table, and inline-table*

Inherits: *no*




```
margin-top: 2em;
margin-right: 250px;
margin-bottom: 1em;
margin-left: 4em;
```

NOTA: Gli elementi inline ignorano il margin top e il margin bottom

- Margin che collassano: collassano il margin top e il bottom in uno solo, non avviene ciò se abbiamo a che fare con elementi float o absolute

PROP. DEL BOX : OVERFLOW

overflow

Values: visible | hidden | scroll | auto | inherit

Default: visible

Applies to: *block-level elements and replaced inline elements (such as images)*

Inherits: *no*

visible

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely cover the remainder of the glass in heavy paper. Any exposed areas (even inside) will get scratched by the flying sand, so it has to be a good seal.

hidden

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also need to completely

scroll

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each

auto *(short text)*

Applying the masks to the glasses is the most labor-intensive part of the process.

auto *(long text)*

Applying the masks to the glasses is the most labor-intensive part of the process. Not only do you have to measure, place, and burnish on each mask, but you also

BORDER

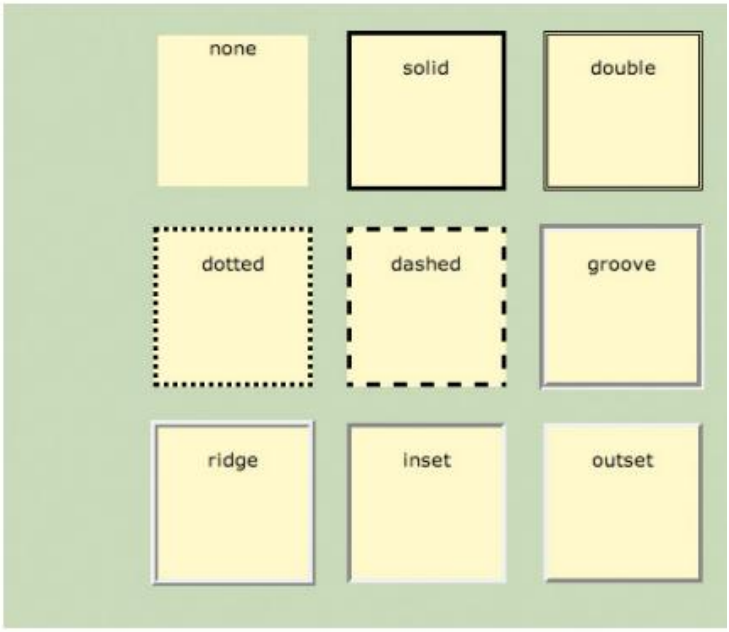
Del contorno di un elemento potete specificare:

- border-style
- border-width
- border-color

Stile del border:

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

Values:	none | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit
Default:	none
Applies to:	*all elements*
Inherits:	*no*



border-style

Values:	none | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit
Default:	none
Applies to:	*all elements*
Inherits:	*no*

Dimensione del border:

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

Values:	*length units* | thin | medium | thick | inherit
Default:	medium
Applies to:	*all elements*
Inherits:	*no*

border-width

Values:	*length units* | thin | medium | thick | inherit
Default:	medium
Applies to:	*all elements*
Inherits:	*no*


```
div#help {
  border-top-width: thin;
  border-right-width: medium;
  border-bottom-width: thick;
  border-left-width: 12px;
  border-style: solid;
  width: 300px;
  height: 100px;
}
div#help {
  border-width: thin medium thick 12px;
  border-style: solid;
  width: 300px;
  height: 100px;
}
```

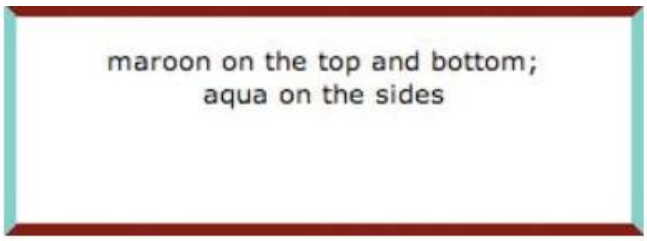
Colore del border:

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

Values:	*color name or RGB value* | transparent | inherit
Default:	*the value of the color property for the element*
Applies to:	*all elements*
Inherits:	*no*

border-color

Values:	*color name or RGB value* | transparent | inherit
Default:	*the value of the color property for the element*
Applies to:	*all elements*
Inherits:	*no*




```
div#special {
  border-color: maroon aqua;
  border-style: solid;
  border-width: 6px;
  width: 300px;
  height: 100px;
}
```

Singole dichiarazioni border:

border-top, border-right, border-bottom, border-left

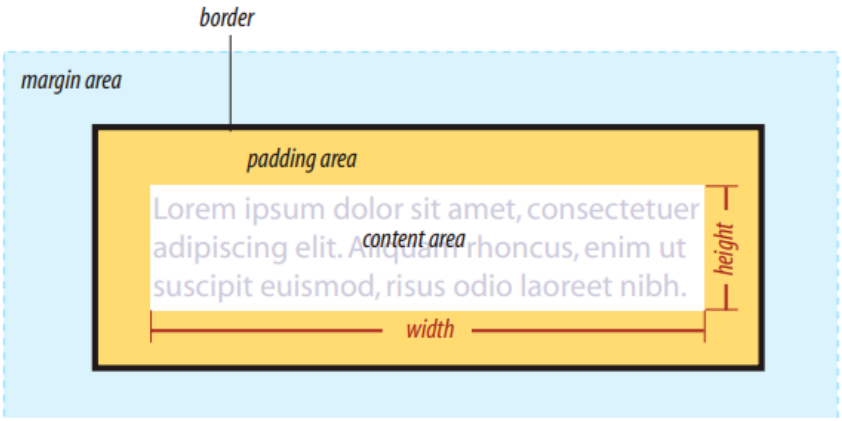
Values:	*border-style border-width border-color* | inherit
Default:	*defaults for each property*
Applies to:	*all elements*
Inherits:	*no*

border

Values:	*border-style border-width border-color* | inherit
Default:	*defaults for each property*
Applies to:	*all elements*
Inherits:	*no*

```
h1 { border-left: red .5em solid; }      /* left border only */
h2 { border-bottom: 1px solid; }         /* bottom border only */
p.example { border: 2px dotted #663; }   /* all four sides */
```

ESEMPIO DI SIZING




```
p {
  background: #c2f670;
  width: 500px;
  height: 150px;
  padding: 20px;
  border: 2px solid gray;
  margin: 20px;
}
```

20px + 2px + 20px + 500px width + 20px + 2px + 20px = 584 pixels

PROPRIETA' BOX-SIZING

Tale proprietà può assumere due valori:

1) **content-box**: il valore di width fa riferimento all'area del contenuto
2) **border-box**: il valore di width fa riferimento al box nella sua interezza, comprendendo cioè anche il padding e il bordo.

GESTIONE DEL BACKGROUND

➔ background-color

È possibile scegliere il colore di sfondo di un elemento con la proprietà background-color.

```
body {
 background-color: lightblue;
}
```

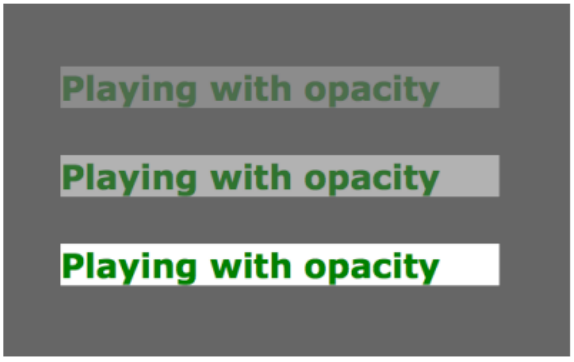
➔ opacity

opacity

NEW IN CSS3

Values:	*number (0 to 1)*
Default:	1
Applies to:	*all elements*
Inherits:	*no*

```
h1 {color: green; background: white; opacity: .25;}
h1 {color: green; background: white; opacity: .5;}
h1 {color: green; background: white; opacity: 1;}
```



➔ Immagini nel background

È possibile scegliere un'immagine di sfondo con la proprietà background-image.
Di default l'immagine verrà ripetuta per riempire l'intero spazio.

```
body {
 background-image: url("utente.png");
}
```

background-repeat: È possibile utilizzare la proprietà di ripetizione background-repeat per un'immagine di sfondo. Generalmente viene utilizzata per riempire l'elemento. Può essere in verticale, orizzontale o non esserci.

```
body {
 background-image: url("utente.png");
 background-repeat: repeat-x; // o repeat-y e no-repeat
}
```

background-position:
È possibile scegliere la posizione di un'immagine con la proprietà background-position.

```
body {
 background-image: url("utente.png");
 background-repeat: no-repeat;
 background-position: right top;
}
```

background-attachment (per gestione dello scroll):

background-attachment

Values:	**scroll** | **fixed** | **local** *(new in CSS3)* | **inherit**
Default:	**scroll**
Applies to:	*all elements*
Inherits:	*no*

PROPRIETA' BACKGROUND

background

Values:	*background-color background-image background-repeat background-attachment background-position* | inherit
Default:	*see indiviual properties*
Applies to:	*all elements*
Inherits:	*no*

LEZ6 – 26/03/2024

CAMBIARE IL DISPLAY

display

Values:	inline|block|list-item|inline-block|table|inline-table| table-row-group|table-header-group|table-footer-group|table-row| table-column-group|table-column|table-cell|table-caption|none *The following are new in CSS3:* run-in|compact|ruby|ruby-base|ruby-text| ruby-base-container|ruby-text-container
Default:	inline
Applies to:	*all elements*
Inherits:	*yes*

STILI PER LISTE

➔ Tipo di lista:

list-style-type

Values: none | disc | circle | square | decimal | decimal-leading-zero | lower-alpha | upper-alpha | lower-latin | upper-latin | lower-roman | upper-roman | lower-greek | inherit

Default: disc

Applies to: ul, ol, *and* li *(or elements whose display value is* list-item*)*

Inherits: *yes*

disc
- crimson
- cobalt
- veridian
- umber
- ultramarine

circle
- crimson
- cobalt
- veridian
- umber
- ultramarine

Keyword	System
decimal	1, 2, 3, 4, 5...
decimal-leading-zero	01, 02, 03, 04, 05...
lower-alpha	a, b, c, d, e...
upper-alpha	A, B, C, D, E...
lower-latin	a, b, c, d, e... (same as lower-alpha)
upper-latin	A, B, C, D, E... (same as upper-alpha)
lower-roman	i, ii, iii, iv, v...
upper-roman	i, ii, iii, iv, v...
lower-greek	α, β, γ, δ, ε...

➔ Posizione marker:

list-style-position

Values: inside | outside | inherit

Default: outside

Applies to: ul, ol, *and* li *(or elements whose display value is* list-item*)*

Inherits: *yes*

Outside
- crimson
- cobalt
- veridian
- umber
- ultramarine

Inside
- crimson
- cobalt
- veridian
- umber
- ultramarine

➔ Immagini per marker:

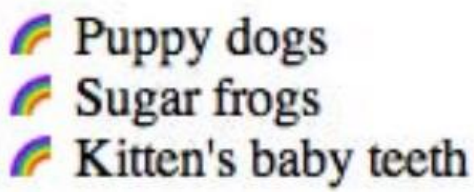
list-style-image

Values: *url* | none | inherit

Default: none

Applies to: ul, ol, *and* li *(or elements whose display value is* list-item*)*

Inherits: *yes*




```
ul {
   list-style-image: url(/images/happy.gif);
   list-style-type: circle;
   list-style-position: outside;
}
```

ANCORA

➔ Punti specifici di una pagina

Identificazione dell'ancora: ogni id può essere usato come ancora.

<h1 id="startH">H</h1>

Creazione del link: si usa come # ed il nome dell'identificatore

… F | G | H | I | J …

➔ Punti specifici di altre pagine

Si aggiunge alla url, #nome identificatore.

Esempio: See the Glossary, letter H

Se l'id non è presente funziona comunque come un link normale.

ATTRIBUTO TARGET

Specifica in quale finestra e dove aprire il link (target="_blank" dice di aprire il link in una nuova finestra).

Esempio: O'Reilly

Oss: Posso dare un nome ad una finestra e usarla più volte.

ALTRI POSSIBILI LINK

Link a mail: Loreti

Link a numeri di telefono: Loreti

STILE DEI LINK

➔ Specifici dei link:

:link per link non visitati

:visited per link già visitati dall'utente (cache del browser)

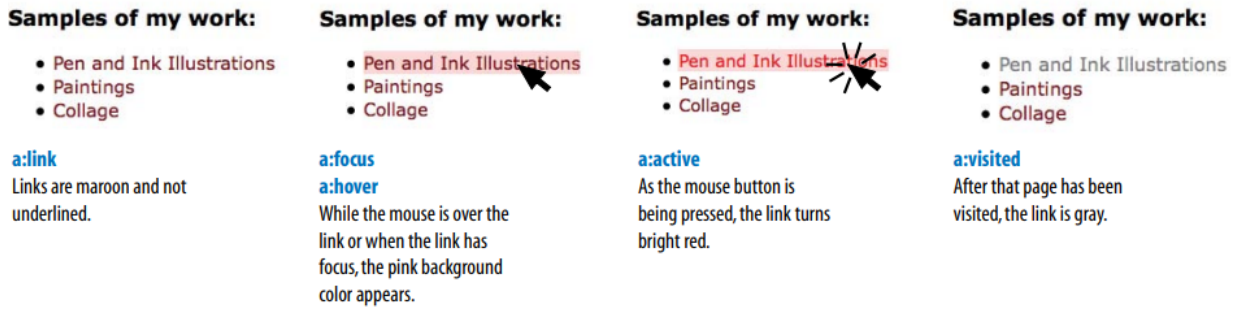
➔ Legati ad azioni dell'utente

:focus se l'elemento è selezionato

:hover se il mouse è sopra l'elemento (deve essere dichiarato dopo :link e :visited)

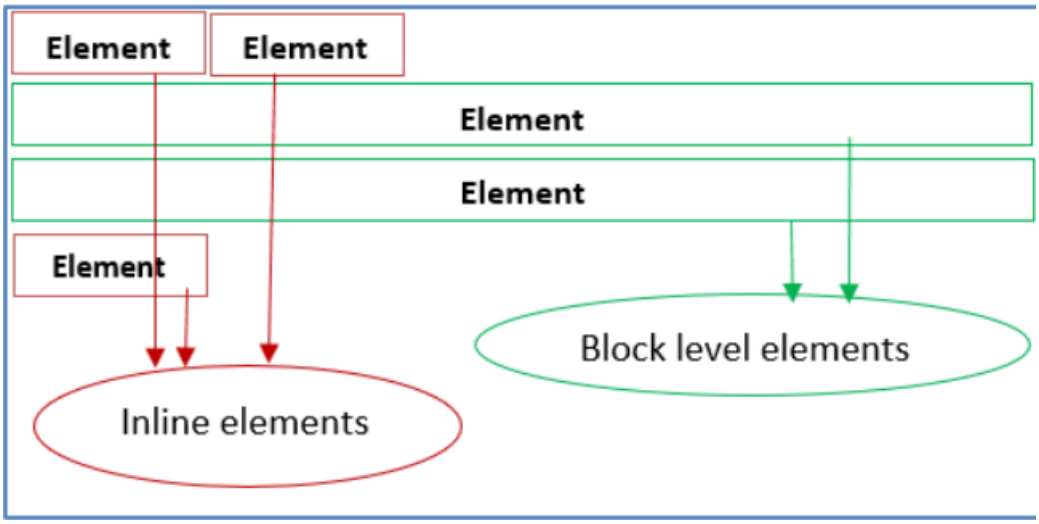
:active per il momento del click (deve essere dichiarato dopo :hover)

```
a { text-decoration: none; }  /* turns underlines off for all links */
a:link { color: maroon; }
a:visited { color: gray; }
a:focus { color: maroon; background-color: #ffd9d9; }
a:hover { color: maroon; background-color: #ffd9d9; }
a:active { color: red; background-color: #ffd9d9; }
```



LEZ6 – 26/03/2024

POSIZIONE DEGLI ELEMENTI



Position per cambiare il flusso:
- top, bottom, left, right
- z-index

PROPRIETA' FLOAT: muove un elemento tutto a destra o tutto a sinistra permettendo agli atri elementi di circondarlo. Questa proprietà può avere i seguenti valori float:
- left - L'elemento fluttua a sinistra del relativo contenitore
- right - L'elemento fluttua a destra del suo contenitore
- none - L'elemento non fluttua (verrà visualizzato solo nel punto in cui si trova nel testo). Questa è l'impostazione predefinita
- inherit - L'elemento eredita il valore float del suo elemento padre

PROPRIETA' POSITION

position è la proprietà fondamentale per la gestione della posizione degli elementi. I valori con cui è possibile definire la modalità di posizionamento sono quattro:

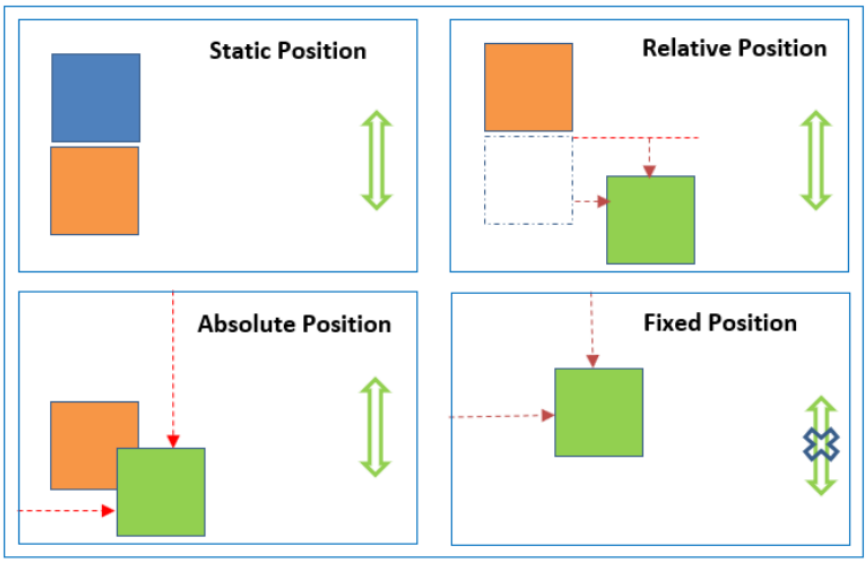
1. static
2. relative
3. absolute
4. fixed

1) Posizione static
   Gli elementi HTML di default sono posizionati statici.
   Gli elementi non posizionati secondo alcun metodo.
   Rappresenta la posizione normale che ciascuno di essi occupa nel flusso del documento.

2) Posizione relative
   L'elemento viene posizionato relativamente al suo box contenitore.
   La posizione viene impostata con le proprietà top, left, bottom o right.
   Nel posizionamento relativo esse non indicano un punto preciso, ma l'ammontare dello spostamento in senso orizzontale e verticale rispetto al box contenitore.

3) Posizione absolute
   L'elemento, o meglio, il box dell'elemento, viene rimosso dal flusso del documento ed è posizionato in base ai valori forniti con le proprietà top, left, bottom o right.

Non dipende dall'elemento in cui è contenuto.
Il posizionamento assoluto (position: absolute;) avviene sempre rispetto al box contenitore dell'elemento. Questo è rappresentato dal primo elemento padre che abbia un posizionamento diverso da static.



SPECIFICARE LA POSIZIONE

`top, right, bottom, left`

Values:	*length measurement* | *percentage* | `auto` | `inherit`
Default:	`auto`
Applies to:	*positioned elements (where position value is* `relative`, `absolute`, *or* `fixed`*)*
Inherits:	*no*

- Specifica la distanza rispetto all'**elemento o blocco contenitore**

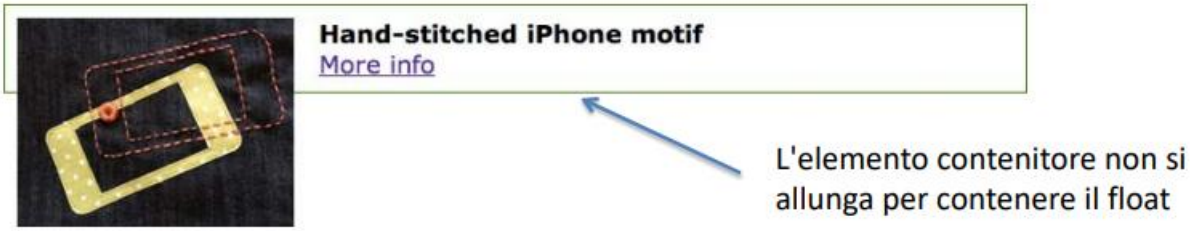
- **Containing block:** primo antenato con position **non static** (o il body)

CLEAR

Per impedire che un elemento successivo subisca anche esso il float di quello precedente si utilizza la proprietà clear.
Visto che il float sposta un elemento dal flusso normale del documento, è possibile che esso venga a trovarsi in posizioni non desiderate, magari al fianco di altri elementi che vogliamo invece tenere separati; clear risolve questo problema.
Può assumere i seguenti valori:

- none → gli elementi con float possono stare a destra e sinistra dell'elemento.
. left → si impedisce il posizionamento a sinistra.
- right → si impedisce il posizionamento a destra.
- both → si impedisce il posizionamento su entrambi i lati.

CONTENERE I FLOAT



Nell'elemento contenitore devo dichiarare overflow auto o hidden

```
#container {
  overflow: auto;
  width: 100%;
  background-color: #GGG;
  padding: 1em;
  }
```

PROPRIETA' DISPLAY

Questa proprietà ci consente di modificare il modo predefinito in cui viene visualizzato qualcosa. Tutto nel flusso normale ha un valore di visualizzazione predefinito. Ad esempio, il fatto che i paragrafi si mostrino uno sotto l'altro è dovuto al fatto che hanno proprietà display: block.

FLEXBOX DISPLAY

Progettato per semplificare la disposizione di elementi in una sola dimensione, sia come riga che come colonna. Per usare la flexbox, bisogna applicare la proprietà display: flex all'elemento padre; tutti i suoi figli diretti diventano quindi oggetti flex.
Il flexbox dispone gli elementi lungo l'asse delle x o delle y automaticamente.

```
.container {
   display: flex;
}
```

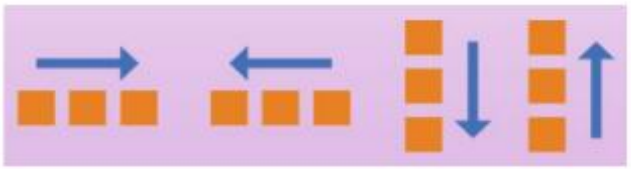
FLEX DIRECTION

Flexbox fornisce una proprietà chiamata flex-direction che specifica in quale direzione viene eseguito l'asse principale (in quale direzione sono disposti i figli della Flexbox): di default, questa è impostata su row (riga), che li rende disposti in fila nella direzione predefinita del browser (da sinistra a destra, nel caso di un browser italiano.

```
.container {
flex-direction: row | row-reverse | column | column-reverse;
}
```

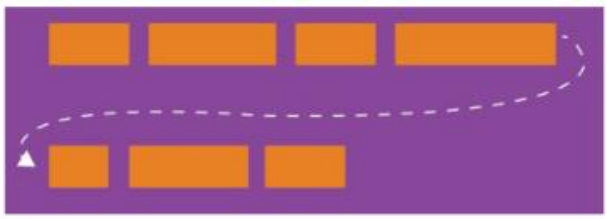
I possibili valori sono:

row (default): da sinistra a destra
row-reverse: da destra a sinistra
column: da sopra a sotto
column-reverse: da sotto a sopra



FLEX WRAP

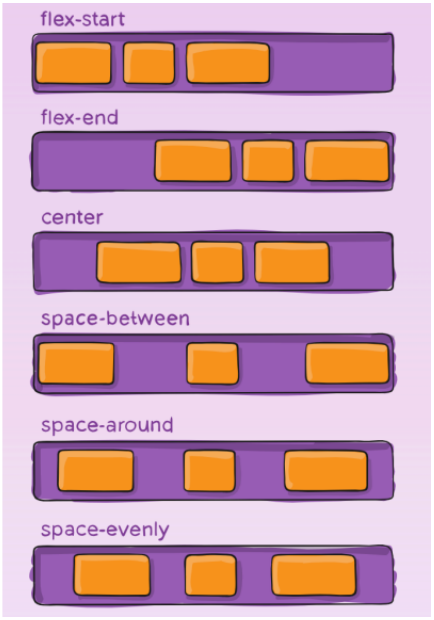
Di default, gli elementi flex cercano di disporsi in una sola riga. Quando gli elementi sono troppi meno spazio avranno a disposizione. Un modo in cui è possibile risolvere questo problema è aggiungere la proprietà **flex-wrap: wrap**



JUSTIFY CONTENT

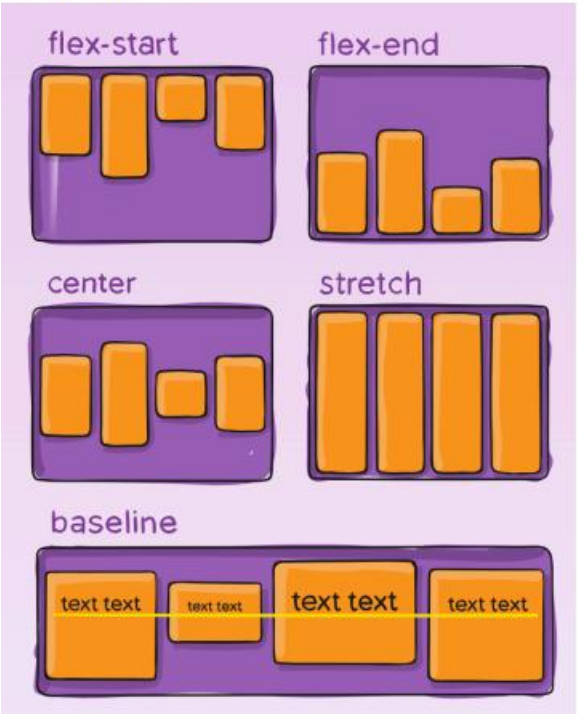
justify-content

```
.container {
        justify-content: ...
}
```

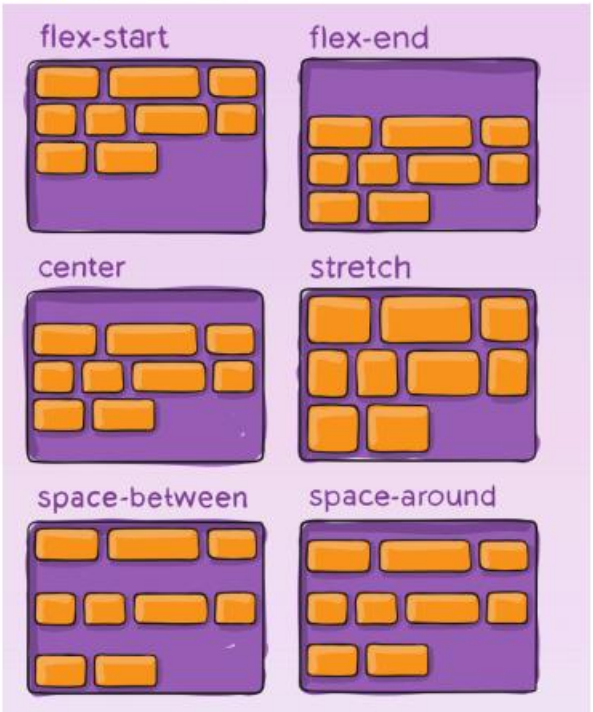


ALIGN-ITEM & ALIGN-CONTENT

align-items:



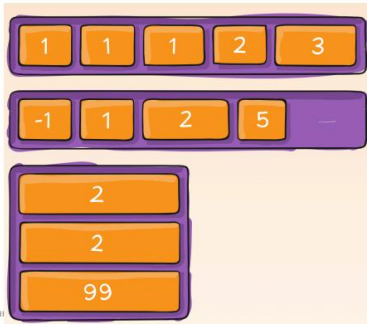
align-content:



PROPRIETA' ORDER

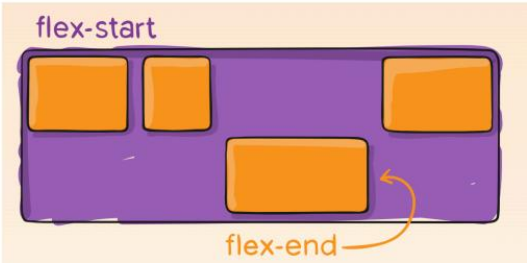
```
div#myRedDIV   {order: 2;}
div#myBlueDIV  {order: 4;}
div#myGreenDIV {order: 3;}
div#myPinkDIV  {order: 1;}
```

```
.item {
    order: <integer>; /* default is 0 */
}
```



PROPRIETA' ALIGN-SELF

```
.item {
    align-self: auto | flex-start | flex-end | center |
baseline | stretch;
}
```



LEZ7 – 28/03/2024 (layout e media query)

LAYOUT DELLE PAGINE

- FIXED (dimensione in pixel fissa)
  Vantaggi:
  - Controllo del numero delle righe
  - è semplice da realizzare
  - è una visualizzazione comune per i desktop

  Svantaggi:
  - se la finestra è piccola la parte destra viene coperta
  - se lo schermo è grande ho spazi a dx ed sx da gestire
  - se ho carattere grandi ho problemi di righe corte
  - la pagina non è controllabile dall'utente
- FLUID (proporzionale alla larghezza del browser)
  Vantaggi:
  - Sembra più vicino alla natura dei browser
  - Elimina spazi potenzialmente vuoti
  - L'utente può controllare la pagina
  - Non ho scrollbar orizzontali

  Svantaggi:
  - Se il monitor è grande ho righe troppo lunghe (limitare la dimensione)
  - Non è facile predire la posizione degli elementi e l'effetto grafico finale
  - Problemi nei browser piccoli
  - I conti sono un poco più complicati
- ELASTIC (proporzionale al testo)
- HYBRID (misto tra fixed e fluid)

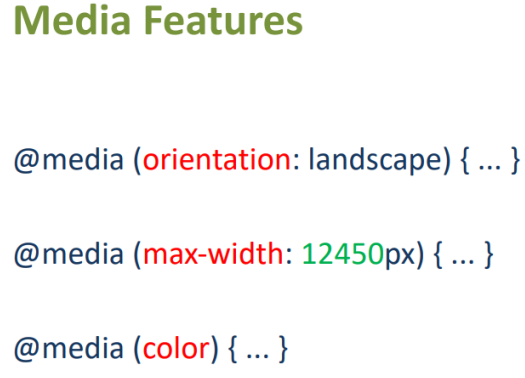
MEDIA QUERY

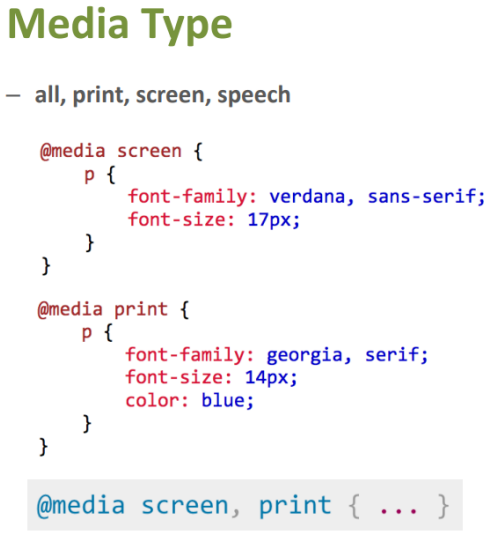
Sono dichiarazioni CSS con le quali è possibile identificare il tipo di dispositivo o una sua caratteristica allo scopo di applicare stili o condizioni differenti in base ad un elenco di regole. Per le loro caratteristiche sono uno strumento molto utilizzato quando si sviluppa in ottica Responsive. Sintassi delle media query:

```
@media not|only mediatype and (media feature) {
    CSS-Code;
}
```

```
<link rel="stylesheet" media="mediatype and|not|only (media feature)" href="mystylesheet.css">
```

Le media query possono essere di più tipi e sono composte principalmente da due elementi, il media-type e il media-features.





➔ Tipi di media query: La tipologia (media-type) indica la categoria di un dispositivo e se non viene espressamente indicata verrà usata la corrispondenza più generica possibile.

All: indica tutti i dispositivi;

Print: è destinata alla modalità di stampa, quindi modificherà un documento o una pagina nella sua versione stampabile.

Screen:
Per gli schermi, i tablet o gli smartphone.

Speech:
Dedicata ai sintetizzatori vocali o per gli screen reader che leggono la pagina ad alta voce.

➔ Media features: le media-features sono dichiarazioni utilizzate nelle Media Queries che consentono di intercettare particolari caratteristiche o "stati" del dispositivo utilizzato.

width La larghezza esatta dell'area di visualizzazione

height L'altezza esatta dell'area di visualizzazione

min-width La larghezza minima dell'area di visualizzazione

min-height L'altezza minima dell'area di visualizzazione

max-width La larghezza massima dell'area di visualizzazione

max-height L'altezza massima dell'area di visualizzazione

orientation L'orientamento del dispositivo (landscape o portrait per dispositivi mobili)

➔ Modalità di utilizzo: un esempio di modalità elementare di utilizzo può essere la manipolazione della visibilità e dello stile degli elementi in base alla max-width del dispositivo

```
@media all and (max-width: 736px) {...}
```

➔ Operatori logici: Gli operatori logici sono utilizzati per strutturare una Media Queries che rispetta più di una condizione alla volta. Ne esistono di quattro tipi:

and Combina più condizioni che devono essere rispettate nello stesso momento

not Per creare una regola di esclusione (è necessario indicare il media-type)

only Applica le regole solo se esiste una corrispondenza diretta (è necessario indicare il media-type)

, (virgola) È utilizzata per combinare più Media Queries in un'unica regola; ogni dichiarazione, separata dalla virgola, verrà analizzata singolarmente (quindi basta che una delle condizioni sia vera)

Esempi:

```
@media (min-width: 30em) and (orientation: landscape) { ... }

@media (min-height: 680px), screen and (orientation: portrait) { ... }
```

LEZ8 – 04/04/2024 (Dispositivi mobili e responsività)

Normalmente i device con lo schermo piccolo tendono a restringere la pagina per adattarla allo schermo

- spesso il testo viene ridotto troppo e non si legge
- i link sono piccoli da cliccare
- gestire lo zoom non sempre fornisce una esperienza piacevole

I siti Responsive forniscono layout diversi a seconda della dimensione della finestra di visualizzazione.

- Un solo documento per tutti i device ma con stile variabile a seconda dello schermo

TECNICHE RESPONSIVE

Gli elementi per rendere un sito responsive sono i seguenti:

1. Controllo viewport
2. Controllo layout con media queries
3. Media "fluidi"

CONTROLLO VIEWPORT

Tag meta da inserire nell'head html è il seguente:

<meta name="viewport" content="width=device-width,initial-scale=1">

Opzioni:

– width, height -> device-width o device-height oppure px
– user-scalable -> no, yes
– initial-scale -> 1
– minimum-scale -> 1 non scala
– maximum-scale -> 1 non scala

With the viewport meta tag

Without the viewport meta tag

BREAKPOINTS

I breakpoint sono punti su una scala ideale di larghezza del viewport in cui si verifica una qualche modifica al layout della pagina.

I breakpoint si definiscono con valori numerici nelle media query

```
@media only screen and (max-width: 600px) {
    body{
        background-color: red;
    }
}

@media only screen and (min-width: 601px) and (max-width: 900px) {
    body{
        background-color: skyblue;
    }
}
```

(valori breakpoints in verde)

Esempi:

```
@media (min-width: 400px) {
  html { background: red; }
}
@media (min-width: 600px) {
  html { background: green; }
}
@media (min-width: 800px) {
  html { background: blue; }
}
```

(override)

```
@media (max-width: 400px) {
  html { background: red; }
}
@media (min-width: 401px) and (max-width: 800px) {
  html { background: green; }
}
@media (min-width: 801px) {
  html { background: blue; }
}
```

(exclusive)

```
html { background: red; }

@media (max-width: 600px) {
  html { background: green; }
}
```

(Desktop first)

```
html { background: red; }

@media (min-width: 600px) {
  html { background: green; }
}
```

(Mobile first)

"MOBILE FIRST" MEDIA QUERIES

- Si parte definendo prima gli stili per i device mobili (si usa min-width)
- Si aggiungono stili e/o sovrascrivono proprietà per quelli più larghi
- La maggior parte dei framework sono mobile first

```
@media (min-width: 768px) {
  .container {
    width: 750px;
  }
}

@media (min-width: 992px) {
  .container {
    width: 970px;
  }
}

@media (min-width: 1200px) {
  .container {
    width: 1170px;
  }
}
```

MEDIA FLESSIBILI

Controllo della larghezza

- width: L'immagine si espande al 100%
- max-width: Se il container è più grande della dimensione in pixel dell'immagine si ferma il resize
- non posso impostare width ed height

La stessa tecnica si può usare per video, object o altri elementi.
Problema: immagini grandi occupano molta banda, il sito risulta lento.

Background-size può assumere i valori responsivi:

- contain

- cover

Ottimizzazione:

```
/* For devices smaller than 400px: */
body {
    background-image: url('img_smallflower.jpg');
}

/* For devices 400px and larger: */
@media only screen and (min-device-width: 400px) {
    body {
        background-image: url('img_flowers.jpg');
    }
}
```

<source> srcset Attribute

Posso specificare immagini diverse a seconda della MQ:

```
<picture>
  <source media="(min-width: 650px)" srcset="img_pink_flowers.jpg">
  <source media="(min-width: 465px)" srcset="img_white_flower.jpg">
  <img src="img_orange_flowers.jpg" alt="Flowers" style="width:auto;">
</picture>
```

LEZ9 – 09/04/2024 (CSS VARIABLES, GRID LAYOUT E BOOTSTRAP)

CUSTOM PROPERTIES

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}

body { background-color: var(--blue); }

h2 { border-bottom: 2px solid var(--blue); }

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
}
```

VARIABILI IN CSS

var(<custom-property-name>,<declaration-value>?)

```
:root {
  --main-bg-color: pink;
}

body {
  background-color: var(--main-bg-color);
}
```

NOTA: Le variabili globali vengono definite nell'elemento root, come segue

```
:root {
  --blue: #6495ed;
  --white: #faf0e6;
}
```

NOTA: Le variabili locali vengono dichiarate in un selettore e posso usarle nei discendenti

```
.alert {
  --alert-color: #ff6f69;
}
```

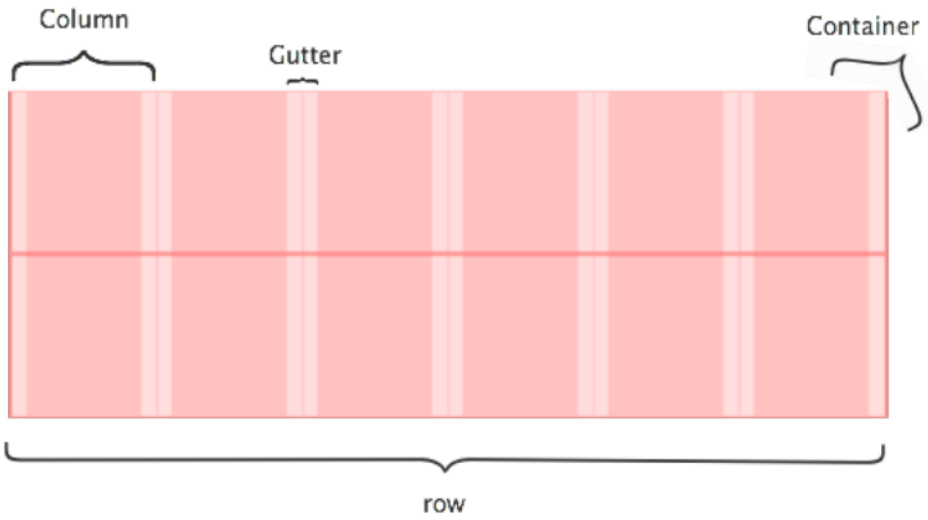
```
.alert p {
  color: var(--alert-color);
  border: 1px solid var(--alert-color);
}
```

NOTA: Posso riscrivere il valore di una variabile in base alle media query

```css
:root {
  --main-font-size: 16px;
}

media all and (max-width: 600px) {
  :root {
    --main-font-size: 12px;
  }
}
```

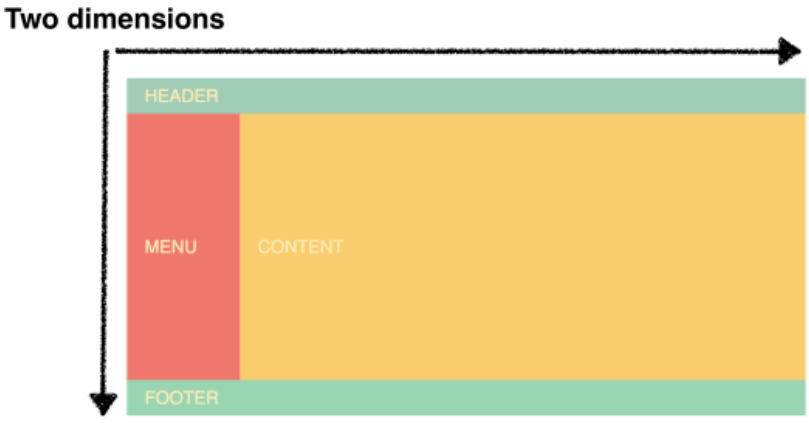
GRID LAYOUT

Struttura:



GRID SYSTEM CSS

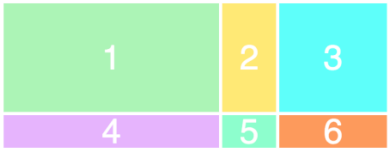
Grid è un sistema bidimensionale



Grid template:

```css
.wrapper {
  display: grid;
  grid-template-columns: 200px 50px 100px;
  grid-template-rows: 100px 30px;
}
```

```html
<div class="wrapper">
  <div>1</div>
  <div>2</div>
  <div>3</div>
  <div>4</div>
  <div>5</div>
  <div>6</div>
</div>
```

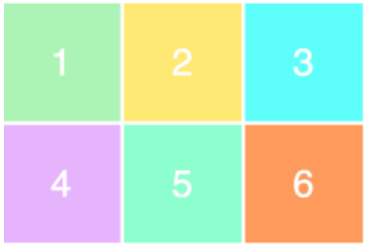


Alcuni esempi di grid template e utilizzo di diverse unità di misura:

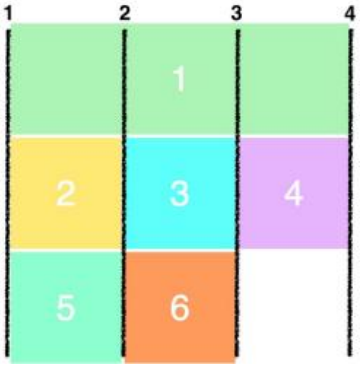
```
grid-template-rows: 320px auto 320px;
grid-template-rows: repeat(3, 275px);
grid-template-rows: 1fr 2fr 1fr;                //fr="frazione"
grid-template-columns: 25% 75%;
grid-template-columns: 200px auto minmax(80px,auto)
```

Posizionamento:

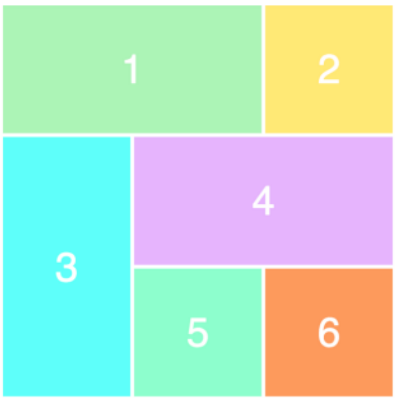
```
.wrapper {
  display: grid;
  grid-template-columns: 100px 100px 100px;
  grid-template-rows: 100px 100px 100px;
}
```



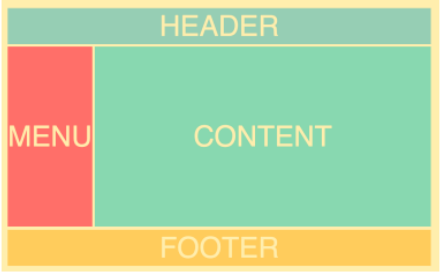

```
.item1 {
        grid-column-start: 1;
        grid-column-end: 4;
}
```



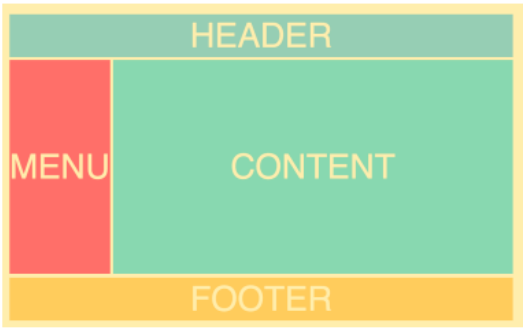

```
.item1 {
        grid-column-start: 1;
        grid-column-end: 3;
}

.item3 {
        grid-row-start: 2;
        grid-row-end: 4;
}
.item4 {
        grid-column-start: 2;
        grid-column-end: 4;
}
```




```
.wrapper{
  display: grid;
  grid-gap: 3px;
  grid-template-columns: repeat(12, 1fr);
  grid-template-rows: 40px 200px 40px;
}

header{ grid-column:  1/  -1;}
nav{ }
main{    grid-column:  2/  -1;}
footer{   grid-column:  1  /  -1;}
```



```
.wrapper{
   display: grid;
   grid-gap: 3px;
   grid-template-columns: repeat(12, 1fr);
   grid-template-rows: 40px 200px 40px;
  grid-template-areas:  "h h h h h h h h h h h h"
                        "m c c c c c c c c c c c"
                        "f f f f f f f f f f f f";

}

header{  grid-area: h;}
nav{     grid-area: m;}
main{    grid-area: c;}
footer{  grid-area: f;}
```



BOOTSTRAP (GRID FRAMEWORK)

- Facile da usare
  - Chiunque abbia solo una conoscenza di base di HTML e CSS può iniziare a utilizzare Bootstrap
- Funzionalità reattive
  - Il CSS reattivo di Bootstrap si adatta a telefoni, tablet e Desktop
- Approccio mobile-first
  - In Bootstrap 3, gli stili mobile-first fanno parte del framework di base
- Compatibilità con i browser
- Bootstrap è compatibile con tutti i browser moderni

Grid Bootstrap:

	Extra small devices Phones (<768px)	**Small devices** Tablets (≥768px)	**Medium devices** Desktops (≥992px)	**Large devices** Desktops (≥1200px)
Grid behavior	Horizontal at all times	Collapsed to start, horizontal above breakpoints		
Container width	None (auto)	750px	970px	1170px
Class prefix	`.col-xs-`	`.col-sm-`	`.col-md-`	`.col-lg-`
# of columns	12			
Column width	Auto	~62px	~81px	~97px
Gutter width	30px (15px on each side of a column)			
Nestable	Yes			
Offsets	Yes			
Column ordering	Yes			

LEZ17 – 09/05/2024 (Javascript Asincrono)

In CSS, i **layout** e le **posizioni** sono strumenti fondamentali per organizzare e controllare la disposizione degli elementi all'interno di una pagina web. Ecco una spiegazione dettagliata di ciascuno:

1. **Layout in CSS**
Il layout in CSS si riferisce alla disposizione degli elementi su una pagina. Esistono diversi metodi per creare layout complessi e flessibili, ognuno con le proprie caratteristiche:

A. **Box Model**
Il **box model** è il modello di base di layout in CSS. Ogni elemento HTML è considerato una scatola (box), che include:
- **Content** (contenuto)
- **Padding** (spazio tra il contenuto e il bordo)
- **Border** (bordo intorno all'elemento)
- **Margin** (spazio esterno rispetto agli elementi vicini)

B. **Flexbox**
Il **Flexible Box Layout** (Flexbox) è un metodo utilizzato per distribuire gli elementi in una singola dimensione (orizzontale o verticale). È molto utile per creare layout dinamici e flessibili, soprattutto per allineare, ridimensionare e ordinare gli elementi.

- **Contenitore flessibile:** Definito con `display: flex;`.
- **Caratteristiche principali:**
  - Distribuisce lo spazio tra e all'interno degli elementi flessibili.
  - Gli elementi possono essere ordinati e allineati facilmente con proprietà come `justify-content`, `align-items`, e `flex-wrap`.

Esempio Flexbox:
```css
.container {
  display: flex;
  justify-content: center; /* Allinea gli elementi orizzontalmente */
  align-items: center; /* Allinea gli elementi verticalmente */
}
```

C. **Grid Layout**
Il **CSS Grid Layout** è un sistema di layout bidimensionale che consente di organizzare gli elementi sia in righe che in colonne. È ideale per creare layout di pagine più complessi.

- **Contenitore grid:** Definito con `display: grid;`.
- **Caratteristiche principali:**
  - Permette di creare layout strutturati con righe e colonne.
  - Gli elementi possono essere posizionati in aree specifiche della griglia tramite coordinate (`grid-column`, `grid-row`).

Esempio Grid:
```css
.container {
  display: grid;
```

```
  grid-template-columns: repeat(3, 1fr); /* 3 colonne di uguale larghezza */
  grid-template-rows: 100px auto; /* 2 righe: la prima fissa, la seconda flessibile */
}
```

D. **Float Layout**
Storicamente, il **float** era usato per creare layout flottanti, ma è diventato meno comune con l'introduzione di Flexbox e Grid.
- **Proprietà float:** Sposta gli elementi a sinistra o a destra, permettendo agli elementi di avvolgersi attorno ad esso.
- **Chiaro con clearfix:** Gli elementi flottanti possono causare problemi di layout se non vengono "puliti" correttamente.

E. **Positioning Layout**
Utilizzando il sistema di **posizionamento** (dettagliato sotto), si può avere un controllo molto più preciso su dove gli elementi appaiono all'interno della pagina.

---

2. **Posizioni in CSS**
La **posizione** in CSS definisce dove e come un elemento viene posizionato all'interno del layout della pagina. Esistono diversi tipi di posizionamento:

A. **Position: Static**
- **Default** per tutti gli elementi.
- Gli elementi con `position: static;` sono posizionati in base al normale flusso del documento (l'ordine degli elementi nell'HTML).

B. **Position: Relative**
- Gli elementi con `position: relative;` vengono posizionati **rispetto alla loro posizione normale** nel flusso del documento.
- Utilizza proprietà come `top`, `right`, `bottom`, `left` per spostare l'elemento dal suo posto originale.
- Gli altri elementi della pagina non vengono influenzati dal suo spostamento.

Esempio:
```css
.elemento {
  position: relative;
  top: 20px; /* Sposta l'elemento 20px verso il basso dalla sua posizione originale */
}
```

C. **Position: Absolute**
- Gli elementi con `position: absolute;` vengono posizionati rispetto al loro **antenato posizionato** più vicino (cioè, un elemento con `position` diversa da `static`).
- Se non c'è un antenato posizionato, l'elemento viene posizionato rispetto alla finestra del browser (viewport).
- Gli elementi "absolute" vengono rimossi dal flusso normale del documento.

Esempio:
```css
.elemento {
  position: absolute;
  top: 10px;
  left: 50px;
}
```

D. **Position: Fixed**
- Gli elementi con `position: fixed;` sono **fissi** rispetto alla finestra del browser, quindi non si muovono quando si scorre la pagina.
- Vengono posizionati in base alla finestra (viewport), utilizzando proprietà come `top`, `right`, `bottom`, `left`.

Esempio:
```css
.header {
  position: fixed;
  top: 0;
  width: 100%; /* Fa in modo che l'elemento si estenda per tutta la larghezza della finestra */
}
```

E. **Position: Sticky**
- Gli elementi con `position: sticky;` si comportano come `relative` finché non raggiungono un certo punto di scroll, dopodiché diventano `fixed`.
- Sono utili per creare intestazioni o menu di navigazione che rimangono visibili durante lo scorrimento.

Esempio:
```css
.menu {
  position: sticky;
  top: 0;
}
```

Riepilogo delle differenze:
- **Static:** Posizione normale senza modifiche.
- **Relative:** Posizionato rispetto alla sua posizione originale.
- **Absolute:** Posizionato rispetto all'elemento posizionato più vicino.
- **Fixed:** Posizionato rispetto alla finestra del browser e non si muove con lo scrolling.
- **Sticky:** Si comporta come `relative` finché non viene raggiunto un certo punto di scrolling, poi diventa `fixed`.

Differenza tra **Layout** e **Posizioni**:

- Il **layout** riguarda la disposizione degli elementi all'interno della pagina in relazione agli altri elementi, spesso utilizzando strumenti come Flexbox o Grid.
- Le **posizioni** definiscono esattamente dove appare un elemento, indipendentemente dal layout complessivo, e permettono un controllo più preciso.

Con questi strumenti puoi creare layout dinamici e flessibili e gestire il posizionamento degli elementi con precisione.