Web Design

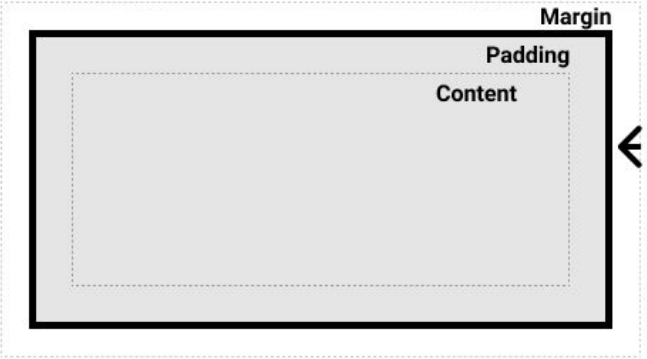
prof.ssa Cristina Iurissevich
cristina.iurissevich@abacatania.it

Indice lezione:

1. CSS Layout
    a. Box Model
    b. Layout
    c. Flexbox
    d. Grid Layout
2. Esercizio in aula
3. Esercizio a casa

CSS Layout

Box Model



Content

L'area in cui vengono visualizzati i contenuti; può essere dimensionato utilizzando le proprietà width e height

Padding

Il padding si trova attorno al contenuto come spazio vuoto.

Border

Il bordo che avvolge il contenuto e l'eventuale spaziatura interna

Margin

Il margine è il livello più esterno che avvolge contenuto, riempimento e bordo e li distanzia da altri elementi

```css
.box {
  width: 350px;
  height: 150px;
  margin: 10px;
  padding: 25px;
  border: 5px solid black;
}
```

Lo spazio effettivo occupato dal box sarà largo 410px (350 + 25 + 25 + 5 + 5) e alto 210px (150 + 25 + 25 + 5 + 5).

Background

La proprietà **background-color** definisce il colore di sfondo su qualsiasi elemento nei CSS. La proprietà accetta qualsiasi <colore> valido. Un colore di sfondo si estende sotto il contenuto e il riquadro di riempimento dell'elemento.

```
.box {
    background-color:red;
}
```

Background-image / Background-repeat

La proprietà **background-image** consente la visualizzazione di un'immagine sullo sfondo di un elemento.

```
.box {
  background-image: url(star.png);
  background-repeat: no-repeat;
}
```

La proprietà **background-repeat** viene utilizzata per controllare il comportamento dell'affiancamento delle immagini. I valori disponibili sono:

- **no-repeat**: impedisce la ripetizione completa dello sfondo.
- **repeat-x**: ripeti orizzontalmente.
- **repeat-y**: ripeti verticalmente.
- **repeat**: l'impostazione predefinita; ripetere in entrambe le direzioni.

Background-size

La proprietà **background-size** può assumere valori <length> o <percentage> che servono a ridimensionare l'immagine per adattarla allo sfondo.

```
.box {
  background-image: url(balloons.jpg);
  background-repeat: no-repeat;
  background-size: 100px 10em;
}
```

Possiamo anche utilizzare i valori:

- **cover**: il browser renderà l'immagine abbastanza grande da coprire completamente l'area del riquadro pur mantenendo le proporzioni (parte dell'immagine potrebbe essere tagliata).
- **contain**: il browser renderà l'immagine della dimensione giusta per adattarsi alla casella (potrebbero esserci degli spazi vuoti su entrambi i lati o nella parte superiore e inferiore dell'immagine).

Background-position

La proprietà **background-position** permette di scegliere la posizione in cui appare l'immagine di sfondo. Questo utilizza un sistema di coordinate in cui l'angolo in alto a sinistra della casella è (0,0) e la casella è posizionata lungo gli assi orizzontale (x) e verticale (y).

Possiamo utilizzare valori come **top** e **right**, lunghezze e percentuali oppure un mix delle due.

```
.box {
  background-image: url(star.png);
  background-repeat: no-repeat;
  background-position: 20px 10%;
}

.box {
  background-image: url(star.png);
  background-repeat: no-repeat;
  background-position: top center;
}
```

Border

In genere quando aggiungiamo bordi a un elemento con CSS utilizziamo una proprietà abbreviata che imposta la larghezza, lo stile e il colore del bordo in una riga di CSS.

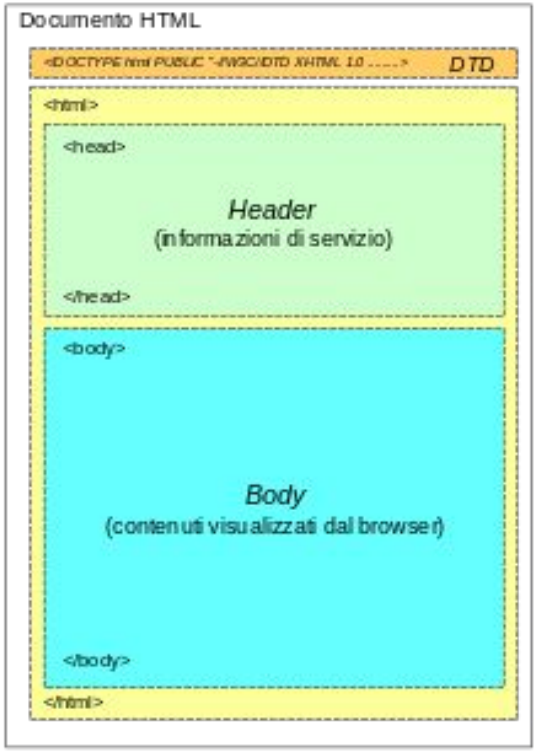
```css
.box {
  border: 1px solid black;
}
```

=

```css
.box {
  border-width: 1px;
  border-style: solid;
  border-color: black;
}
```

Precisazioni

Anatomia



<!DOCTYPE html> dichiara che questo è un foglio html.

<html> rappresenta l'inizio (e la fine del nostro codice).

```
1   <!DOCTYPE html>
2   <html lang="it">
3   >   <head>…
5       </head>
6   >   <body>…
15      </body>
16  </html>
```

In **<head>** abbiamo tutte le informazioni per il browser.

In **<body>** tutto ciò che verrà visualizzato all'interno della finestra del browser.

Elementi <head>

```html
<head>
    <!-- Codifica caratteri -->
    <meta charset="utf-8" />
    <!-- Il titolo visualizzato nella "linguetta" del browser -->
    <title>La mia pagina</title>
    <!-- Collegamento al foglio di stile -->
    <link rel="stylesheet" type="text/css" href="foglio-stile.css" />
    <!-- Aggiunta favicon -->
    <link rel="icon" href="favicon.ico" type="image/x-icon" />
    <!-- Adattabilità alla viewport -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
</head>
```

Ovviamente ce ne sono anche molti altri.

Percorsi

Un collegamento deve portare ad una sola destinazione perciò nel nostro link (attributo href o src) dobbiamo indicare la posizione esatta del collegamento.

Si utilizza un **percorso assoluto** se il collegamento si trova sullo stesso sito web già pubblicato online o su un sito web esterno.

Si utilizza un **percorso relativo** se si trova sullo stesso sito web e vogliamo essere liberi di poter modificare l'indirizzo web.

Percorsi assoluti

In un percorso assoluto si comunica al browser di andare a prendere la destinazione partendo direttamente dal dominio e aggiungendo la posizione del file che si vuole raggiungere.

Esempio:

https://www.abacatania.it/elenco-professori/

```
<p><a href="https://www.abacatania.it/elenco-professori/">Elenco professori</a></p>
```

Percorsi relativi

Può capitare di fare riferimento a documenti situati all'interno del sito e non avere ancora un indirizzo web, oppure di dover spostare un sito da un'indirizzo a un altro e perciò non voler reimpostare da capo tutti i link. In questi casi è necessario utilizzare i percorsi relativi.

I percorsi relativi fanno riferimento alla posizione degli altri file rispetto al documento in cui ci si trova in quel momento.

Per linkare due pagine che si trovano all'interno della stessa directory è sufficiente scrivere:

```
<a href="pagina2.html">Pagina 2</a>
```

/sottodirectory

Per far riferimento a un file contenuto in una cartella di livello inferiore alla posizione corrente (cioé al file a cui vogliamo collegarlo), bisogna nominare la cartella (o le cartelle divise tra loro dallo /) seguita dal carattere **/** , e poi il nome del file.

Secondo la formula: **cartella/nomeFile.estensione**


```
<a href="prima/interna/interna.html">Pagina interna</a>
```

/directory di livello superiore

Se dalla pagina interna vogliamo far riferimento a una pagina (index.html) che si trova più in alto dobbiamo utilizzare la dicitura **../**


```
<a href="../../index.html">Pagina interna</a>
```

Indentatura HTML

```
<!DOCTYPE html>
<html lang="it">
    <head>
        <title>Ciao Mondo!</title>
    </head>
    <body>
        <header>
            <h1>Ciao Mondo!</h1>
        </header>
        <main>
            <section>
                <p>Questa è la nostra prima pagina HTML!</p>
            </section>
        </main>
    </body>
</html>
```

Indentatura CSS

```css
body {
    font-family: 'Roboto', sans-serif;
    text-align: justify;

}
section {
    margin-left: 40px;
}
.parent {
    display: flex;
    flex-direction: row;
    flex-wrap: wrap;
    align-items: center;
}
.parent figure {
    margin: 6px 6px 20px 6px;
}
.parent img {
    width: 430px;
    height: 457px;
    object-fit: cover;
}
```

Commenti

```html
<!DOCTYPE html>
<html lang="it">
    <!-- Questo è un commento, non influisce sul documento ma ci può aiutare -->
    <head>…
    </head>
    <body>…
    </body>
</html>
```

```css
/*Questo è un commento, non influisce sul documento ma ci può aiutare*/
body {
    font-family: 'Roboto', sans-serif;
    text-align: justify;

}
```

I commenti ci servono per segnare determinate note all'interno del nostro file senza andare a intaccare il codice.

In html il commento si scrive: **<!-- -->**

Nel css invece: **/* */**

Piccolo museo virtuale

- Creazione cartella: “museo-virtuale”
- All’interno della cartella create un file: “home.html”
- Scaricate tre immagini di opere d’arte a vostra scelta e rinominatele con nomi semplici (es. “girasoli.jpg”), inseritele in una sottocartella della cartella museo-virtuale nominata: “img”

Homepage

1. Apriamo il file “home.html”
2. Cominciamo a compilare le dichiarazioni (doctype e <html> con i corretti attributi) e i dati necessari in <head>:
    - Decodifica caratteri (UTF-8)
    - Viewport
    - Titolo del sito
3. Inseriamo (sotto l’elemento <head>) l’elemento <body> al cui interno ci dovranno essere 3 sezioni:
    - <header> contenente scritta o logo sito che linka a #
    - <main>
    - <footer> contenente il vostro nome e cognome e la data di oggi
4. All’interno dell’elemento <main> dovranno essere presenti le tre immagini che linkano a tre pagine html (artista1.html - artista2.html - artista3.html).

3 Pagine artista

Nelle 3 pagine html relative ai 3 artisti (artista1.html - artista2.html - artista3.html):

1. Inseriamo tutti gli elementi inseriti nella pagina home.html
2. Inseriamo il link all'homepage sulla scritta o logo presente in <header>
3. Procediamo a modificare solo l'elemento <main> inserendo:
    - nome dell'artista come intestazione (h2),
    - un breve testo descrittivo dell'artista (p),
    - almeno 6 immagini di opere dell'artista
    - un link (a) che permetta di ritornare all'home page

Piccolo museo virtuale

css

- All’interno della cartella “museo-virtuale” inseriamo un nuovo file nominato “stile.css”
- Inserire in tutti i file html il collegamento al foglio di stile nel tag <head>

Stile:
colore, font, border, margin e padding

1. Impostiamo:
    - colore, font e dimensioni del testo (con differenze per intestazioni e paragrafi)
    - colore dei link (a)
    - colore dei link hover (a:hover)
2. Impostiamo colori diversi per le sezioni header, main e footer
3. Le cornici delle immagini (in homepage e nelle pagine degli artisti) devono possedere un bordo, un margine e un padding.

Layout

Le tecniche di impaginazione più comuni sono:

- Flusso normale (normal flow): è il modo in cui il browser dispone le pagine HTML per impostazione predefinita quando non fai nulla per controllare il layout della pagina.
- **Flexbox** (CSS Flexible Box Layout): progettato per semplificare la disposizione delle cose in una dimensione, sia come riga che come colonna. Per usare flexbox si applica "**display: flex**" all'elemento genitore degli elementi che vogliamo disporre; tutti i suoi figli diventano elementi flessibili.
- **Grid Layout**: mentre flexbox è progettato per un layout unidimensionale, Grid Layout è progettato per due dimensioni, allineando le cose in righe e colonne. Come per flexbox, abilitiamo Grid Layout con il suo valore di visualizzazione specifico "**display: grid**".

Flexbox

elemento genitore

elementi figli

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
```

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

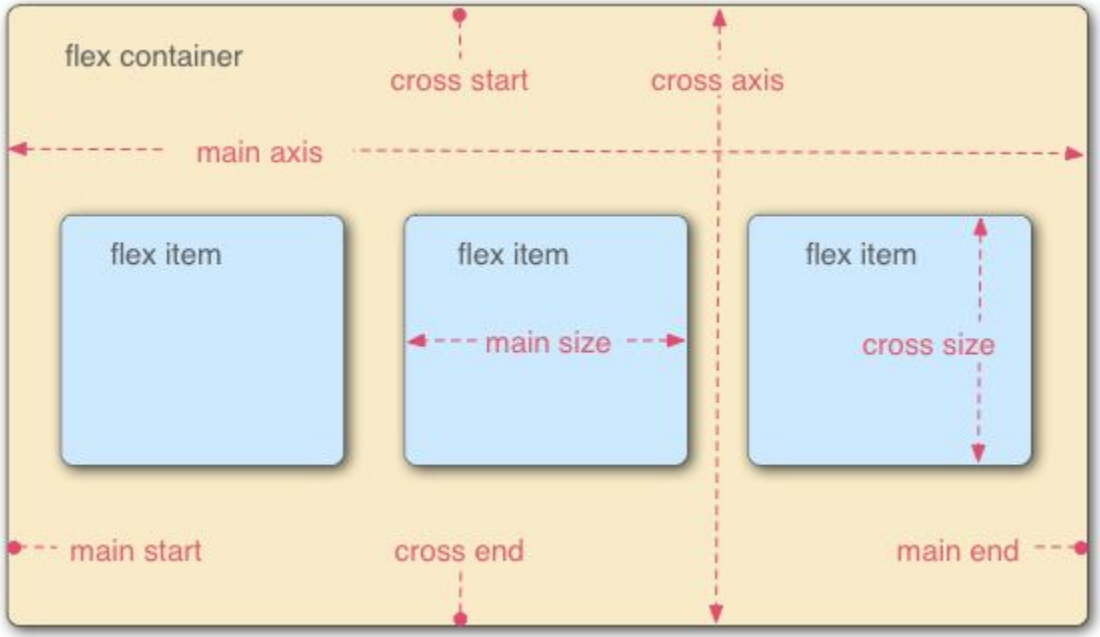
Quando aggiungiamo “display: flex” al genitore, i tre elementi al suo interno si dispongono in colonne. Ciò è dovuto al fatto che diventano elementi flessibili e sono influenzati da alcuni valori iniziali impostati da flexbox sul contenitore flessibile.

https://css-tricks.com/snippets/css/a-guide-to-flexbox/

https://flexboxfroggy.com/#it

https://codepen.io/cristina_iurissevich/pen/ZEZLvZm

Flexbox


flex container
cross start
cross axis
main axis
flex item
flex item
main size
flex item
cross size
main start
cross end
main end

Flexbox / Elemento genitore

Proprietà per gli elementi genitori:

- **flex-direction**: specifica in quale direzione scorre l'asse principale (in quale direzione sono disposti gli elementi interni). Per impostazione predefinita, questo è impostato su riga (row).
- **flex-wrap**: specifica se gli elementi flessibili devono andare a capo oppure no
- **justify-content**: utilizzato per allineare orizzontalmente gli elementi.
- **align-items**: utilizzato per allineare verticalmente gli elementi.

https://css-tricks.com/snippets/css/a-guide-to-flexbox/#aa-properties-for-the-parentflex-container

Flexbox / Elemento figlio

Gli elementi figlio, cioé interni ad un contenitore flessibile diventano automaticamente elementi flessibili (flex). Le proprietà possibili per un elemento figlio sono:

- **order**: specifica l'ordine degli elementi flessibili.
- **flex-grow**: definisce la capacità di un elemento flessibile di prendere più spazio se necessario.
- **flex-shrink**: definisce la capacità di un elemento flessibile di ridurre il suo spazio se necessario.
- **flex-basis**: specifica la lunghezza iniziale di un elemento flessibile.
- **align-self**: consente di sovrascrivere l'allineamento predefinito (o quello specificato da align-items) per i singoli elementi.

https://css-tricks.com/snippets/css/a-guide-to-flexbox/#aa-properties-for-the-childrenflex-items

Piccolo museo virtuale

Flexbox

Nella pagina artista2.html utilizziamo i flexbox per gestire le immagini inserite:

1. flex-direction: row
2. flex-wrap: wrap
3. justify-content: center
4. align-items: flex-start

Grid Layout

CSS Grid non è un concorrente di Flexbox.

```
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
```

Possono interagire e collaborare in layout complessi, perché CSS Grid lavora in 2 dimensioni (righe e colonne), mentre Flexbox lavora in una sola dimensione (righe o colonne).

Il layout Grid viene attivato sull'elemento genitore impostando **display: grid**.

Come per flexbox, possiamo definire alcune proprietà sul contenitore e alcune proprietà su i singoli elementi della griglia. La combinazione di queste proprietà determina l'aspetto finale della griglia.

https://css-tricks.com/snippets/css/complete-guide-grid/

https://cssgridgarden.com/#it

https://codepen.io/cristina_iurissevich/pen/BaEpYam

grid-template-columns

elemento genitore

elementi figli

```html
<div class="wrapper">
  <div class="box1">One</div>
  <div class="box2">Two</div>
  <div class="box3">Three</div>
</div>
```

```css
.wrapper {
  display: grid;
}
```

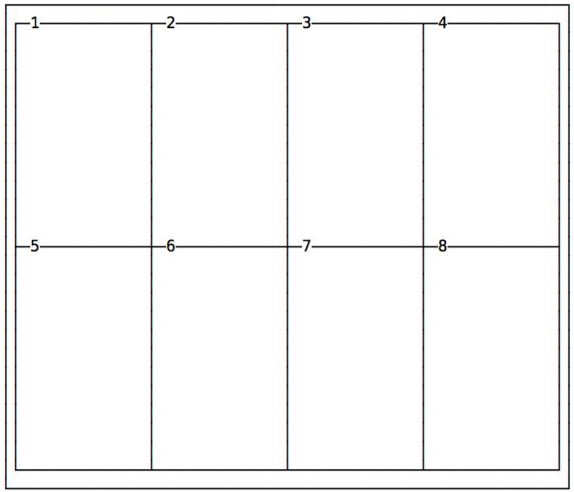
Quando aggiungiamo “**display: grid**” al genitore, a differenza di Flexbox, gli elementi figli non appariranno immediatamente diversi. Gli elementi continueranno a essere visualizzati uno sotto l'altro come fanno nel flusso normale. Per vedere qualcosa dovremo aggiungere alcune colonne alla griglia.

```css
.wrapper {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
}
```

grid-template-columns / grid-template-rows

Queste due proprietà definiscono il numero di colonne (**grid-template-columns**) e righe (**grid-template-rows**) della griglia, e ne regolano anche la larghezza di ogni colonna e riga.

```
.wrapper {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  grid-template-rows: 300px 300px;
}
```



Nell'esempio abbiamo utilizzato valori uguali per tutte le righe e tutte le colonne a ovviamente possiamo anche utilizzare valori differenti.

Es: grid-template-columns: 200px 100px 300px 50px;

```
.wrapper {
  display: grid;
  grid-template-rows: 300px auto 300px;
}
```

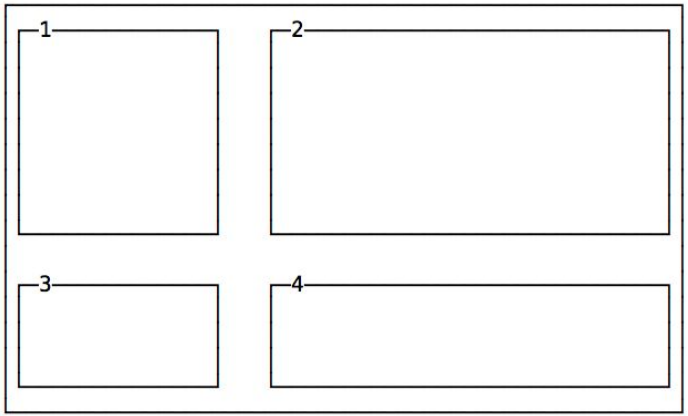
Possiamo anche utilizzare il valore auto per chiedere a quella “cella” di adattarsi alla grandezza del suo contenuto.

Nell’esempio definiamo 3 righe di cui la prima e la terza hanno un’altezza fissa di 300px e quella intermedia si adatterà in altezza al suo contenuto.

grid-column-gap / grid-row-gap

Se non vengono specificati dei valori non ci sarà spazio tra le celle.

```
.wrapper {
  display: grid;
  grid-template-columns: 100px 200px;
  grid-template-rows: 100px 50px;
  grid-column-gap: 25px;
  grid-row-gap: 25px;
}
```



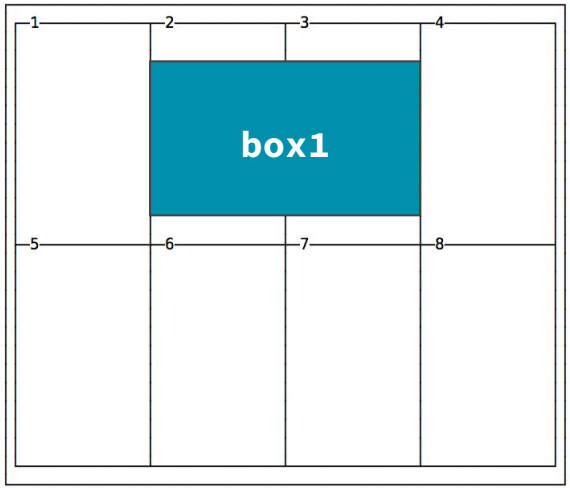
Possiamo anche descrivere lo stesso layout con la scorciatoia:

grid-gap: 25px;

grid-column-start / grid-column-end
grid-row-start / grid-row-end

Ogni elemento ha la possibilità di occupare più di una cella nella riga ed espandersi orizzontalmente o verticalmente, rispettando le proporzioni che abbiamo creato per la griglia.

```
.wrapper {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  grid-template-rows: 100px 50px;
}
.box1 {
  grid-column-start: 2;
  grid-column-end: 4;
}
```

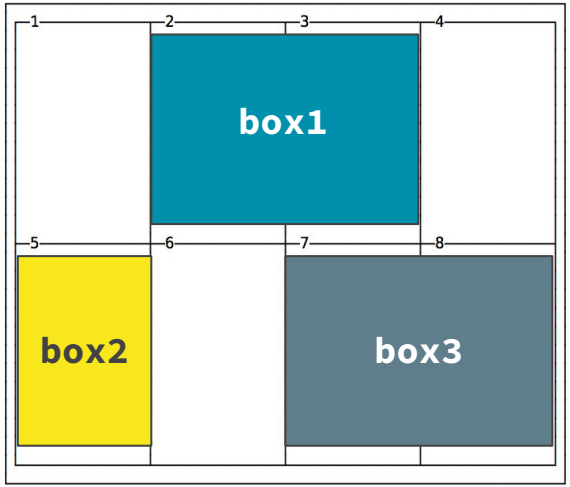


Lo stesso principio si applica per le righe (grid-row-start e grid-row-end).

grid-template-areas

Possiamo impostare una cella vuota usando un punto **.** invece di un nome di area in grid-template-areas:

```
.wrapper {
  display: grid;
  grid-template-columns: 200px 200px 200px 200px;
  grid-template-areas:
      ". header header ."
      "sidebar . main main";
}
.box1 {
  grid-area: header;
}
.box2 {
  grid-area: sidebar;
}
.box3 {
  grid-area: main;
}
```



Piccolo museo virtuale

Grid

Nella pagina artista3.html utilizziamo il Grid layout per gestire le immagini inserite:

1. xxxx

Esercizio a casa

1. Su https://www.freecodecamp.org/ continuiamo la certificazione “Responsive Web Design” facendo i moduli:
    a. “**Learn the CSS Box Model by Building a Rothko Painting**”
    b. “**Learn CSS Flexbox by Building a Photo Gallery**”
    c. “**Learn CSS Grid by Building a Magazine**”