Web Design

prof.ssa Cristina Iurissevich
cristinaiurissevich@abacatania.it

Layout

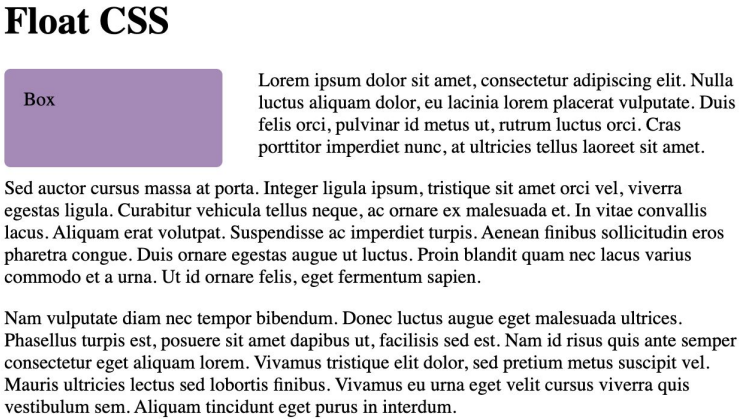
Float

— — —

La proprietà float è stata introdotta per implementare layout che coinvolgono un'immagine fluttuante all'interno di una colonna di testo, con il testo che si avvolge a sinistra o a destra di essa.

Rimane possibile rendere mobile qualsiasi cosa, non solo le immagini.

La proprietà float indica a un elemento di "fluttuare" nella direzione specificata. All'immagine in questo esempio viene indicato di fluttuare in alto verso sinistra per poter "avvolgere" gli elementi di pari livello. Puoi indicare a un elemento di galleggiare left, right, none o inherit (eredita il valore dell'elemento genitore).



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

Clear

Quando utilizziamo la proprietà float e vogliamo che l'elemento successivo sia sotto (non a destra o a sinistra), dovremo utilizzare la proprietà clear che specifica cosa dovrebbe accadere con l'elemento che è accanto a un elemento fluttuante.

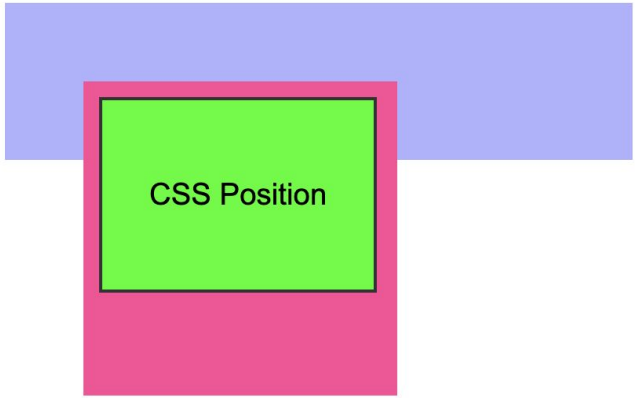
La proprietà clear può avere uno dei seguenti valori:

none (valore predefinito), left, right, both (spostato sotto sia gli elementi fluttuanti a sinistra che quelli a destra), inherit (eredita il valore di clear dal suo elemento genitore).

Il tuo elemento fluttuante continuerà a fluttuare, ma l'elemento cancellato apparirà sotto di esso sulla pagina web.

Posizionamento

Il posizionamento è ciò che determina dove gli elementi appaiono sullo schermo e come appaiono. Possiamo spostare gli elementi in giro e posizionarli con la proprietà CSS position.



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

La proprietà position specifica il tipo di metodo di posizionamento utilizzato per un elemento.

- **static** - posizionato in base al flusso normale della pagina. Gli elementi statici non sono influenzati dalle proprietà top, bottom, left e right.
- **relative** - è posizionato rispetto alla sua posizione normale, ma attraverso le proprietà top, right, bottom e left potrà spostarsi dalla sua posizione normale.
- **fixed** - è posizionato rispetto alla viewport, perciò rimane fermo anche se la pagina viene scrollata. Le proprietà top, right, bottom e left vengono utilizzate per posizionare l'elemento.
- **absolute** - è posizionato rispetto al parente posizionato più vicino. Se non ha genitori posizionati, utilizza l'elemento <body> come riferimento. Gli elementi posizionati in modo assoluto vengono rimossi dal flusso normale e possono sovrapporsi agli elementi.
- **sticky** - è posizionato in base alla posizione di scorrimento dell'utente. Viene posizionato in modo relativo fino a quando si raggiunge la posizione (conferita con top, bottom, left e right) nella finestra di visualizzazione, quindi rimane "appiccicato" in quel punto (come position:fixed).

Position: relative

Un elemento con position: relative è posizionato rispetto alla sua posizione normale.

L'impostazione delle proprietà top, right, bottom e left di un elemento posizionato in modo relativo farà sì che sia spostato dalla sua posizione normale.

```
h1.relative {
  position: relative;
  left: 20px;
  border: 5px solid yellow;
}
```

Position: fixed

Un elemento con position: fixed; è posizionato rispetto alla finestra di visualizzazione, perciò rimane presente anche se la pagina viene scrollata.

Le proprietà top, right, bottom e left vengono utilizzate per posizionare l'elemento.

```
h1 {
  position: fixed;
  bottom: 0;
  left: 0;
  width: 300px;
  border: 5px solid yellow;
}
```

Position: absolute

Un elemento con position: absolute è posizionato rispetto all'elemento posizionato più vicino. Tuttavia, se un elemento posizionato in modo assoluto non ha genitori posizionati, utilizza l'elemento <body> come riferimento.

Gli elementi posizionati in modo assoluto vengono rimossi dal flusso normale e possono sovrapporsi agli elementi.

```css
h1 {
  position: relative;
  width: 400px;
  height: 400px;
  border: 5px solid yellow;
}

h2 {
  position: absolute;
  top: 80px;
  right: 0;
  width: 400px;
  height: 400px;
  border: 5px solid red;
}
```

Position:sticky

Un elemento con position: sticky è posizionato in base alla posizione di scorrimento dell'utente.

Un elemento sticky alterna tra il posizionamento relativo e fisso, a seconda della posizione di scorrimento. Viene posizionato in modo relativo fino a quando si raggiunge una determinata posizione nella finestra di visualizzazione, quindi rimane fissato in quel punto (come position:fixed).

```
div.sticky {
  position: sticky;
  top: 0;
  background-color: black;
  border: 5px solid red;
}
```

Esercizio 6

Posizionamento

Crea una pagina HTML con un elemento main con all’interno:

- Un div con posizione "absolute" e posizionalo a 30px dal bordo superiore destro della viewport.
- Un div con posizione “fixed” e posizionalo rispetto al bordo inferiore destro della finestra del browser con una distanza di 20px.
- Un div con posizione "sticky" dentro un contenitore (un altro div con larghezza 200px che contiene oltre al div sticky abbastanza testo sia sopra che sotto il div sticky per poter farlo scorrere) Il valore del div sticky è top: 50px.

Display

La proprietà display di un oggetto determina se e come verrà presentato nel browser.

Può utilizzare i seguenti valori (qui sono elencati solo i principali):

- **block** - gli elementi sono impilati gli uni sugli altri e ogni elemento occupa il 100% della pagina.
- **inline** - valore predefinito di display per tutti gli elementi in CSS. Gli elementi inline non hanno alcun margine o padding applicati. Lo stesso vale per altezza e larghezza.
- **none** - fa scomparire un elemento. Esiste ancora nell'HTML, ma non è visibile nel browser.
- **flex** - crea un genitore flexbox
- **grid** - crea un genitore grid

Z-index

La proprietà z-index imposta esplicitamente un ordine dei livelli per il codice HTML in base allo spazio 3D del browser, ovvero l'asse Z. Questo è l'asse che mostra i livelli più vicini e lontani da te. L'asse verticale sul Web è l'asse Y, mentre l'asse orizzontale è l'asse X.

La proprietà z-index accetta un valore numerico, che può essere un numero positivo o negativo. Gli elementi verranno visualizzati sopra un altro elemento se hanno un valore z-index più alto.

Nel flusso normale, **se imposti un valore specifico per z-index e questo non funziona devi impostare il valore position dell'elemento su un valore diverso da static**. Questo non accade se ti trovi in un contesto flexbox o grid.

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

Responsive

Il **responsive web design** (RWD) è l'approccio volto a rendere le pagine Web ben visualizzate su schermi di tutte le dimensioni e risoluzioni, garantendo al tempo stesso una buona usabilità. È il modo che si utilizza per progettare un web multi-dispositivo.

Dobbiamo offrire agli utenti mobili un'esperienza adeguata, poiché ci sono ancora vincoli come la durata della batteria e la larghezza di banda.

Il termine "**responsive design**", coniato da Ethan Marcotte nel 2010, descrive l'utilizzo di griglie fluide, immagini fluide e query multimediali per creare contenuti reattivi.

I moderni metodi di layout CSS sono intrinsecamente reattivi ed esistono una moltitudine di funzionalità integrate nella piattaforma web per rendere più semplice la progettazione di siti reattivi.

Desktop

@media screen and
(min-width: 1024px)
{...}

Tablet

@media screen and
(min-width: 768px) and
(max-width: 1023px)
{...}

Smartphone

@media screen and
(max-width: 767px)
{...}

Viewport meta tag

Questo meta tag indica ai browser che devono impostare la larghezza del viewport sulla larghezza del dispositivo e ridimensionare il documento al 100% della dimensione prevista.

Impostando width=device-width stai sovrascrivendo l'impostazione predefinita di un dispositivo mobile (ad esempio per Apple 980px) con la larghezza effettiva del dispositivo.

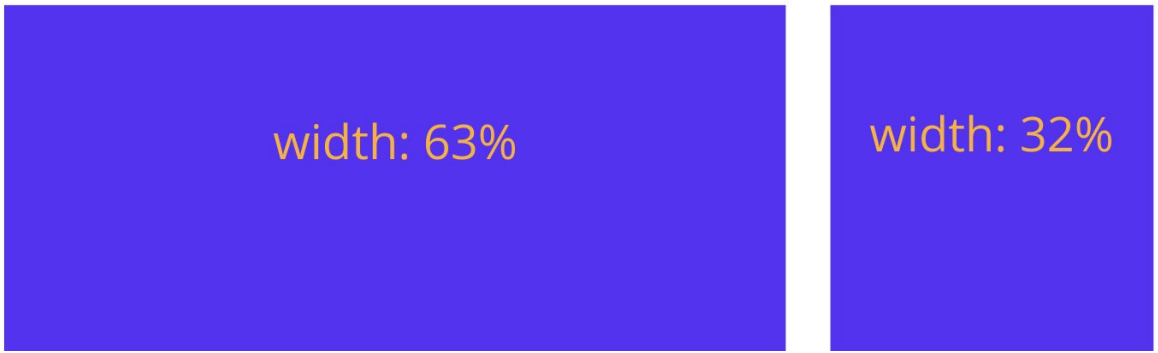
Quindi dovresti sempre includere il meta tag viewport nell'intestazione dei tuoi documenti.

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

Layout fluidi

Un layout fluido si basa su valori dinamici come una percentuale della larghezza del viewport.

Questa tecnica aumenterà o diminuirà dinamicamente le diverse dimensioni degli elementi del contenitore in base alle dimensioni dello schermo.



Layout Flexbox e Grid

Flexbox e Grid sono progettati per essere più efficienti nel disporre più elementi, anche quando la dimensione del contenuto all'interno dei contenitore è sconosciuta.

Un contenitore flessibile espande gli elementi in modo che riempiano lo spazio libero disponibile o li restringe per evitare che trabordino. Questi contenitori flessibili hanno una serie di proprietà uniche, come justify-content, che non è possibile modificare con un normale elemento HTML.

Grid ci da la possibilità di creare griglie differenti a seconda dei breakpoints scelti e creare un'unico codice che si modellerà attraverso le griglie costruite all'inizio.

Immagini reattive

L’iterazione più elementare delle immagini reattive segue lo stesso concetto di layout fluido, utilizzando un’unità dinamica per controllare la larghezza o l’altezza.

L’unità % si avvicina ad una singola percentuale della larghezza o dell’altezza del viewport e fa in modo che l’immagine rimanga proporzionata allo schermo. Il problema di questo approccio è che ogni utente deve scaricare l’immagine a grandezza naturale, anche su cellulare.

```
img {
  max-width: 100%;
}
```

Tipografia Reattiva

Per un design veramente reattivo, si dovrebbero regolare in modo appropriato anche le dimensioni dei caratteri, in modo che si adattino alle dimensioni dello schermo.

Un metodo è impostare un valore statico per la dimensione del font, come 22 px, e adattarlo in ogni media query.

Un'alternativa sono le unità viewport vw che possono essere utilizzate anche per abilitare la tipografia reattiva, senza la necessità di impostare punti di interruzione con le query.

1vw è uguale all'1% della larghezza del viewport, il che significa che se imposti la dimensione del carattere utilizzando vw, si riferirà sempre alla dimensione del viewport.

```
@media (min-width:
1200px) {
    h1 {
   font-size: 22px;
        }
}
```

```
h1 {
  font-size: 6vw;
}
```

Il problema con le unità viewport è che l'utente perde la possibilità di ingrandire qualsiasi testo impostato utilizzando l'unità vw, poiché quel testo è sempre correlato alla dimensione del viewport. Pertanto non dovresti mai impostare il testo utilizzando solo queste unità.

Esiste una soluzione e implica l'utilizzo di calc(). Se aggiungi l'unità vw a un set di valori utilizzando una dimensione fissa come ems o rems, il testo sarà comunque zoomabile.

```
h1 {
  font-size: calc(1.5rem + 3vw);
}
```

Il carattere aumenta quindi gradualmente man mano che si aumenta la dimensione del viewport.

Media Queries

La seguente query verifica se la pagina Web corrente viene visualizzata come supporto su schermo (quindi non come documento stampato) e il viewport è compreso tra 768px e 1023px (dimensioni del tablet). Il CSS per il selettore .container verrà applicato solo se queste due cose sono vere.

```
@media screen and (min-width: 768px)
and (max-width: 1023px){
  .container {
    margin: 10px;
  }
}
```

Le media query ci consentono di applicare i CSS in modo selettivo per modellare la pagina in modo appropriato alle esigenze dell'utente.

Utilizziamo prima la regola base e successivamente la media query.

https://css-tricks.com/a-complete-guide-to-css-media-queries/

Breakpoints

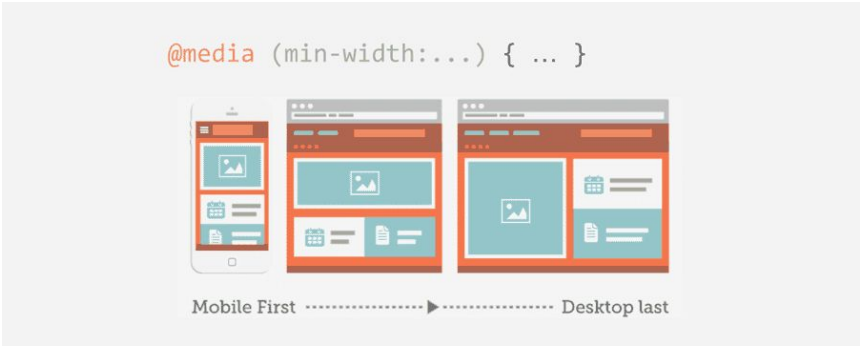
Per lavorare con le media query, è necessario decidere i “**breakpoint responsive**” o i breakpoint delle dimensioni dello schermo. Un breakpoint è la larghezza dello schermo in cui si utilizza una media query per implementare nuovi stili CSS.

Dimensioni comuni dello schermo

- Mobile: 360 x 640
- Mobile: 375 x 667
- Mobile: 360 x 720
- iPhone X: 375 x 812
- Pixel 2: 411 x 731
- Tablet: 768 x 1024
- Laptop: 1366 x 768
- Laptop o desktop ad alta risoluzione: 1920 x 1080

Mobile first

Se si sceglie un approccio di design **mobile-first**, con una singola colonna e font di dimensioni più piccole come base, non è necessario includere i breakpoint mobili – a meno che non si voglia ottimizzare il design per modelli specifici. In questo modo è possibile creare un design reattivo di base con due soli breakpoint, uno per tablet e uno per computer portatili e desktop.



Esercizio 5
Responsive

- Crea una pagina HTML e un foglio CSS collegato.
- Nell'elemento Header inserisci un H1 con un'istanza media query CSS per cambiare il colore del testo del testo in verde quando la larghezza della finestra del browser è inferiore a 600 pixel.
- Dentro all'elemento Main inserisci due immagini (impostate con width: 100%) e applica un'istanza di media query CSS per ridurre le dimensioni dell'immagine a 49% della larghezza originale quando la larghezza della finestra del browser è inferiore a 800 pixel.

Effetti / Animazioni

Animazioni

Esistono due tipologie di animazione:

- **transition** sono animazioni che rispondono soltanto alle interazioni con l'utente e definiscono la trasformazione tra uno stato e un altro. Le transizioni consentono di definire la transizione tra due stati di un elemento. Diversi stati possono essere definiti usando pseudo-classi come :hover o :active.
- **keyframe** possono inserire qualsiasi grado di complessità e passaggi che definiscono l'animazione. Controlla i passaggi intermedi in una sequenza di animazione CSS definendo gli stili per i fotogrammi chiave lungo la sequenza di animazione. Ciò offre un maggiore controllo sui passaggi intermedi della sequenza di animazione rispetto alle transizioni.

Transition

La proprietà transition è composta da:

- **transition-delay**: specifica il tempo di attesa prima di avviare l'effetto di transizione di una proprietà quando il suo valore cambia. Un valore di 0s (o 0ms) inizierà immediatamente l'effetto di transizione. Un valore positivo ritarderà l'inizio dell'effetto di transizione per il periodo di tempo specificato. Un valore negativo avvierà l'effetto di transizione immediatamente, l'effetto verrà animato come se fosse già in esecuzione da un determinato periodo di tempo.
- **transition-duration**: indica il tempo necessario per la transizione. Un tempo pari a 0s indica che non avverrà alcuna transizione, ovvero il passaggio tra i due stati sarà istantaneo. Un valore negativo rende la dichiarazione non valida.
- **transition-property**: imposta le proprietà CSS a cui deve essere applicato un effetto di transizione.
- **transition-timing-function**: descrive come verranno calcolati i valori intermedi utilizzati durante una transizione. Questi effetti sono comunemente chiamati funzioni di andamento.

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

Keyframes

La proprietà keyframes è composta da:

- **from o 0%**: indica la situazione iniziale
- **end o 100%**: indica la situazione finale.
- **% intermedie**: indica la situazione nelle varie posizioni di tempo.

```
@keyframes "animazione" {
  0% {
      /*Effetti inizio animazione*/
      }
  100% {
      /*Effetti fine animazione*/
      }
}
```

```
#box {
  animation-name: animazione;
  animation-duration: 3s;
  animation-iteration-count: 5;
}
```

Assegnamo l'animazione al contenitore e definiamola:

- **animation name**: indica il nome dato al keyframe e lo richiama
- **animation-duration**: indica la la durata totale dell'animazione.
- **animation-iteration-count**: quante volte verrà eseguita l'animazione prima di tornare sullo stato di partenza

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

Filtri

I filtri ci permettono di eseguire delle operazioni sugli elementi, come cambiare l'opacità, la luminosità e altro. La proprietà filter, che può essere applicata ad un'immagine come a qualsiasi altro elemento.

```
img {
  filter: <valore> ;
}
```

Valori possibili:

- blur()
- brightness() -
- contrast()
- drop-shadow()
- grayscale()
- hue-rotate()
- invert()
- opacity()
- sepia()
- saturate()
- url()

Trasformazioni

La proprietà transform accetta queste funzioni:

- **translate()** per spostare elementi
- **rotate()** per ruotare elementi
- **scale()** per scalare elementi in dimensione
- **skew()** per distorcere o inclinare un elemento

Esistono anche le funzioni specifiche per gli assi:

- translateX() o translateY() per spostare gli elementi attorno all'asse X o Y
- scaleX() o scaleY() per scalare le dimensioni di un elemento sull'asse X o Y
- skewX() o skewY() per distorcere o inclinare un elemento sull'asse X o Y

https://css-transform.moro.es/

Esercizio 7

Animazioni

Crea una pagina HTML con un elemento main con all'interno:

- Un Div di dimensioni 200x100 pixel e di colore magenta. Applica una transizione CSS alla posizione del div in modo che cambi gradualmente a 50 pixel verso destra quando il mouse vi passa sopra (:hover).
- Un div di colore blu. Applica un'animazione CSS al colore di sfondo in modo che cambi gradualmente in rosso per poi tornare a blu in un loop infinito.

Web font

Esistono solo poche font garantite da tutti i browser: i cosiddetti caratteri web safe.

È possibile scegliere i caratteri preferiti, seguiti da alternative sicure per il Web e infine dal carattere di sistema predefinito.

Esiste un'alternativa per caricare altri font

I CSS consentono di specificare i file dei caratteri, disponibili sul Web, da scaricare insieme al sito Web. Ciò significa che qualsiasi browser che supporta questa funzionalità CSS può visualizzare i caratteri scelti.

Possiamo utilizzare il nome della famiglia di caratteri specificato all'interno di @font-face per applicare una font personalizzata:

```
@font-face {
      font-family: "myFont";
      src: url("myFont.woff2");
}
html {
  font-family: "myFont", "Arial", sans-serif;
}
```

Google Font

Possiamo anche inserire dei Google Font (https://fonts.google.com/) scegliendo quello che preferiamo e collegandolo al nostro sito utilizzando i link nell'elemento <head>

```
<head>
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/nome_e_pesi_font" rel="stylesheet">
</head>
```