CSS

    - Dimensioni content area
        - Larghezza
        - Altezza
    - Padding
    - Margine
    - Bordo
        - Stile del bordo
        - Dimensione del bordo
        - Colore del bordo
        - Dichiarazione unica
    - Overflow (nel caso qualcosa strabuzza)
    - Sizing
        - Box Sizing
    - Ombra Box
- Background
    - Background color
        - Opacità
    - Background image
    - Background repeat
    - Background position
    - Background attachment
    - Background dichiarazione unica
- Display
    - FLEXBOX Display
        - Direzione del flex
        - Wrap del Flex
        - Giustificazione/spaziatura contenuto
        - Allineamento oggetti
        - Ordine (nel flex)
        - Allinea oggetto singolo
        - Gap tra gli oggetti
- Liste
    - Tipo di lista

    - Marker posizione
    - Immagini per marker
- Posizione degli elementi
    - Posizione
    - Profondità
    - Float
- Layout Delle Pagine
    - Fluid
    - Fixed
    - Responsive Web Design
        - Breakpoints
- CSS Variables
    - Custom Properties
- Grid Layout e Bootstrap
    - Grid
        - Grid Template
        - Posizionamento
            - Grid Template Areas
    - Framework Responsive
        - Bootstrap
- reset.css

CSS

Aggiungere una stylesheet a un documento HTML

Nell'head aggiungere `<link rel="stylesheet" href="nomefile.css">`

Selettori semplici

- Selettore di tipo elemento, seleziona tutti gli elementi:
    - `p {color: red; text-align: center;`
    - `* {}` seleziona ogni elemento
- Selettore classe
    - Seleziona gli elementi che appartengono alla classe

    - `<h1 class="center">Heading</h1>`
    - `class=identificatore`
    - `.` come selettore dell'identificatore
    - `.center {text-align: center, color:red;}` oppure `h1.center {text-align: center, color:red;}` per selezionare gli elementi specifici di una classe
- Selettore ID
    - seleziono l'elemento con un ID **UNIVOCO**
    - preceduto nel CSS da `#`
    - `<p id="para1">...</p>`
    - `#para1 { text-align: center; color:red; }`
- ID per uno solo, classe per più di uno. Non usare ID se non strettamente necessario. Tutti utilizzano le classi.
- è possibile raggruppare selettori
    - `h1, h2, p { color:red; }`

Relazioni nel DOM (Document Object Model)

- Ogni file HTML ha una struttura ad albero con `html` come radice
- **Descendant**: elementi contenuti in un elemento
- **Child**: discendenti diretti (inverso si dice Parent)
- **Ancestor**: elementi che si trovano sopra
- **Parent**: elementi direttamente sopra
- **Sibling**: stesso parent

Selettori composti

- Utilizzano la struttura del DOM
- Utilizzare il segno tra parentesi per selezionare
- Selettori per **descendant** (`spazio`)
    - `p em {color:grey;}`: tutti gli elementi enfatizzati discendenti di un paragrafo (c'è uno spazio tra `p` ed `em`)
- Selettori per **child** (`>`)
    - `div > p {background-color: yellow;}`: tutti i paragrafi contenuti direttamente in un div
- Selettori per **adjacent sibling** (`+`): fratello immediatamente successivo
- Selettori per **general sibling** (`~`): tutti i fratelli successivi

Pseudo Classi

- si chiamano con il `:`
- per definire uno stato speciale di un elemento
  - mouse over
  - visited e unvisited link
  - focus
  - anche per selezionare uno specifico figlio o uno specifico tipo
- `selettore:pseudo-class {property:value;}`
  - `a:visited {color:#00FF00;}`
  - `a:hover {color:#FF0000;}`

Pseudo Elementi

- Dare uno stile alle parte specifiche di un elemento
- CSS genera un elemento dinamicamente, modificando il DOM
- `selector::pseudo-element {property:value;}`
  - `::first-letter`
  - `::after`
  - `::first-line`
  - `::selection`
  - `p::first-letter {color: #FF00000; font-size: xx-large;}`

Selettori con attributi

- Seleziona solo gli elementi che ha un attributo
- `[attribute]`
- `[attribute=value]`
- `img[alt] {background-color: grey;}`
- `a[target="_blank"] {color: red;}`
- è possibile anche selezionare solo parte del valore (parole, inizio, fine, parte specifica) (REGEX)

Ereditarietà

- Alcune proprietà sono ereditate dai discendenti:
  - per esempio `color: value;`: quelle che hanno effetto sui **colori**
  - mentre alcune come `border` no: quelle che hanno effetto sulla **spaziatura**
  - Stili relativi ai **font**
  - Altezza riga
  - Allineamento testo

Conflitti

- è possibile applicare più stili sullo stesso elemento
- alcune proprietà le eredito
- ***di solito*** vince la dichiarazione più vicina all'elemento, **l'ultima**
- più il selettore è specifico più è dominante
    - Selettore base < Classe < ID

Classi multiple

- `class="classe1 classe2 class3..."`
- le classi si "sommano"

!important

- la proprietà non può essere sovrascritta
- `property: value !important;`

Cascade

- algoritmo che definisce come vengono combinate le proprietà che provengono da fonti/stylesheet diverse (browser, author/developer, reader/user)
- esiste un ordine di priorità (dal più basso al più alto)
    1. Browser/ user-agent
    2. Utente
    3. Autore/Sviluppatore
    4. Autore !important
    5. Utente !important
    6. Browser !important

Calcolo della specificità

- a ogni dichiarazione è attribuita una *specificità* misurata con quattro valori [a,b,c,d]
- "a" il più importante "d" il meno importante
- [0,1,0,0] vs [0,0,5,5] vince il primo
    - si misura da sinistra a destra andando avanti se uguali
- "a" 1 se la dichiarazione è inline, 0 altrimenti
- "b" numero di selettori id
- "c" numero di selettori classe, attributo o pseudo-classe
- "d" numero di selettori elemento o pseudo elemento

Formattazione del testo

Font

Font Families

- I caratteri/font si dividono in famiglie
  - **Serif**: quelle con i *ghirigori/fronzoli*
  - **Sans-serif**: quelle senza
  - **Monospace**: per scrivere il codice
- Non tutti i font sono disponibili su ogni macchina, per questo si utilizzano **sequenze di font**, con alla fine il font default del browser (sans-serif, serif, monospace)
- forniamo una lista di "typeface" di applicare al testo: `font-family: lista, di, font, separati, con, virgola;`
- se un font non può essere utilizzato si passa a quello successivo

Web Font

- è possibile caricare e far scaricare Font da fonti hostate localmente o remotamente (e.g. Google Fonts)
  - File hostato localmente: `@font-face { font-family: "PPL"; src: url("ppl.woff2");}`
  - File hostato remotamente `@import url('https://fonts.googleapis.com/css2?family=Jolly+Lodger&display=swap');` o con un `<link href="link" rel="stylesheet">`
- ogni font deve avere più file per il *regular, bold, italic*, e per le varie dimensioni, altrimenti non si potranno utilizzare
  - qualche volta tutto è incluso in un solo file

Dimensione

- `font-size: X;`
  - unità di misura relative:
    - `px`: (unità assoluta ma relativa in versioni passate boh)
    - `em`: unità di misura equivalente alla dimensione del font corrente (base o ereditato), X è un numero anche decimale (e.g. di un `<p>` è 16px) (frazione della dimensione del font)
    - `%`: percentuale relativa all'elemento padre
    - `vw`: unità di misura relativa alla dimensione della larghezza del viewport (1 vw = 1/100 del viewport)
    - `vh`: altezza del viewport
  - unità di misura assolute: `px, pt (points), pc (picas), mm, cm, in` e `small, large, x-large, x-small, etc...`

Peso del font

- `font-weight`
  - `black`, `bold`, `medium`, `regular`, `light`, `thin`

Stile del font

- `font-style`
  - `roman`, `italic`, `condensed`, `condensed italic`

Altezza del rigo

- `line-height`
- insieme a `font-size` è molto importante per la leggibilità
- senza unità di misura è in riferimento alla dimensione del font
- altrimenti `em` e `%`

Dichiarazione combo Font

- `font: [style] [variant] [weight] size[/line-height] family` (tra parentesi facoltativo)

Color

- colore del testo
- modi di specifica:
  - Nome in inglese: `color: grey;`
  - Colori RGB Hex letti a coppie (6 nibble): `color: #666666;`
  - Colori RGB 3 numeri se le coppie hanno lo stesso numero: `color: #666;`
  - Colori RGB con funzione: `color: rgb(110, 230, 0);`
- colori web-safe
- RGB + Canale alfa per la trasparenza: `color: rgpa(0, 0, 0, .4);`, trasparenza da 0 a 1
- Color Hunt

Allineamento del testo

- `text-align`
  - `left`, `right`, `center`, `justify`

Effetti sul testo

- `text-decoration`
  - `none`, `underline`, `overline`, `line-through`
  - si usa spesso per levare il sottolineato ai link `a`

Trasformazione del testo (minuscole maiuscole)

- `text-transform`
  - `none`, `capitalize`, `lowercase`, `uppercase`

Indentazione primo rigo

- `text-indent`
  - misure in `px` e adiacenti, `em`, `%`, anche negativi

Ombra del testo

- `text-shadow: 'offset orizzontale' 'offset verticale' 'raggio di sfocatura' 'colore'`
  - misure in `px` e adiacenti, `em`, `%`, anche negativi
  - è possibile inserire più ombre separandole con una virgola

Spaziatura di lettere e spaziatura di parole

- `letter-spacing`
- `word-spacing`
  - misure in `px` e adiacenti, `em`

Box

- è il rettagonolo che "circonda" ogni elemento HTML
- si può vedere se si applica un `border` a ogni elemento
- **Box Model**
- del Box possiamo modificare:
  - il margine (area tra box)
  - il bordo
  - il padding (area tra contenuto e bordo)
  - area del contenuto (altezza e larghezza)
- è possibile vedere il box model e le parti che lo compongono nell'ispettore del browser

Dimensioni content area

Larghezza

- `width`
  - misure in `px`, `em`, `%`, `auto`, e `inherit`, di default è `auto`

Altezza

- `height`
  - misure in `px`, `em`, `%`, `auto`, e `inherit`, di default è `auto`

Padding

- `padding-top, padding-right, padding-bottom, padding-left`
- `padding` (seque l'ordine scritto sopra)
  - misure in `px`, `em`, `%`, e `inherit`, di default è 0

Margine

- `margin-top`, `margin-right`, `margin-bottom`, `margin-left`
- `margin` (segue l'ordine scritto sopra)
  - misure in `px`, `em`, `%`, `auto`, e `inherit`, di default è `auto`
- Gli elementi inline ignorano il margine top e bottom (possiamo modificare solo quelli left e right), tranne quelli replaced come le immagini
- Quando ci sono due margini che si sovrappongono, si applica il margine più grande (collasso dei margini)
  - non collassano se elemento float o absolute
- Si può impostare un valore negativo per un margine per motivi grafici

Bordo

Stile del bordo

- `border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style`
- `border-style` (segue l'ordine scritto sopra)
  - `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`, `inherit`, default `none`

Dimensione del bordo

- `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`
- `border-width` (neanche lo scrivo)
  - misure `px`, `thin`, `medium`, `thick`, `inherit`, default `medium`

Colore del bordo

- `border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color`
- `border-color`
  - nome colore o valori RGB, `transparent`, `inherit`, default è colore dell'elemento

Dichiarazione unica

- `border-top`, `border-right`, `border-bottom`, `border-left`
- `border`
  - `border-style`, `border-width`, `border-color`, `inherit`

Overflow (nel caso qualcosa strabuzza)

- `overflow`
  - `visible`, `hidden`, `scroll`, `auto`, `inherit`, default `visible`

Sizing

- I pixel di content area e margin, si sommano, orizzontalmente (con width) e verticalmente (con height)
- Di solito ci viene data la largezza delle **colonne** (e.g. barra laterale per pubblicità)
- Il margine non è editabile di norma, non si può colorare

Box Sizing

- `box-sizing`
  - cosa va incluso nella somma dei pixel (sizing)
  - `content-box` (default): somma solo la content area
  - `border-box`: include anche padding e border
    - utilizzato per poter lavorare con percentuali

Ombra Box

- `box-shadow`
  - `none`, `offset-orizzontale` `offset-verticale` [`sfocatura`] [`spread`] [`colore`] [`inset`]
    - `offset-orizzontale`: valore positivo ombra a destra, negativo sinistra
    - `offset-verticale`: valore positivo ombra sotto, negativo sopra
    - `sfocatura`: più è alto più è sfocato
    - `spread`: più è alto più è grande l'ombra
    - `inset`: cambia l'ombra da una ombra esterna a una interna (si scrive inset e basta)
  - si possono mettere più ombre separandole da una virgola

Background

Background color

- `background-color`
  - colore in inglese o valore numerico

Opacità

- `opacity`
  - numero da 0 a 1

Background image

- `background-image`
  - `url=(...)`: link immagine, `none`, `inherit`

Background repeat

- `background-repeat`
  - `repeat`, `repeat-x`, `repeat-y`, `no-repeat`, `inherit`

Background position

- `background-position`
  - misure in pixel, percentuali, `left`, `center`, `right`, `top`, `bottom`, `inherit`
  - si possono combinare
  - ordine: width - height

Background attachment

- `background-attachment`
  - `scroll` (default), `fixed`, `local`, `inherit`
  - `fixed` si può usare per fare effetti carini sui siti
  - `local` muove lo sfondo in base al contenuto dell'elemento

Background dichiarazione unica

- `background`
  - `background-color background-image background-repeat background-attachment background-position`, `inherit`

Display

- cambia il tipo di display che ha un elemento (e.g. elemento inline, elemento block)
- `inline-block`: fa diventare un pochino blocco un elemento inline, cioè sarà possibile modificare il margine top e bottom

FLEXBOX Display

- `display: flex;`

1. la prima proprietà che comanda i figli, di solito su CSS le proprietà si applicano a se stessi
2. tl.dr Tutti gli elementi all'interno diventano colonne

Direzione del flex

- `flex-direction`
  - `row` (default), `row-reverse`, `column`, `column-reverse`, `initial`, `inherit`
    - `row`: orizzontalmente, come riga
    - `row-reverse`: auto esplicativo
    - `column`: verticalmente, come colonna
    - `column-reverse`: auto esplicativo

Wrap del Flex

- `flex-wrap`
  - `nowrap` (default), `wrap`, `wrap-reverse`
  - decidere se gli elementi wrappano (quando raggiungono la fine del contenitore "vanno a capo" al posto di restringersi)

Giustificazione/spaziatura contenuto

- `justify-content`
  - `flex-start` (default), `flex-end`, `center`, `space-between`, `space-around`, `space-evenly`
  - allinea gli oggetti flex al centro del contenitore
    - `flex-start`: oggetti posizionati all'inizio del contenitore
    - `flex-end`: fine del contenitore
    - `center`: centro del contenitore
    - `space-between`: spazio tra gli oggetti
    - `space-around`: spazio prima, tra, e dopo gli oggetti
    - `space-evenly`: gli oggetti avranno spazio equale attorno a loro
  - asse orizzontale

Allineamento oggetti

- `align-items` (agisce sulle singole righe o colonne)
  - `normal` (default), `flex-start`, `flex-end`, `center`, `stretch`, `baseline`
    - `normal`: simile a `stretch`
    - `flex-start`: oggetti posizionati all'inizio del contenitore
    - `flex-end`: fine del contenitore
    - `center`: centro del contenitore

        - `stretch`: oggetti sono stretchati per riempire il contenitore
        - `baseline`: oggetti sono posizionati sulla baseline (tutto il testo sarà sulla stessa "riga")
    - asse verticale
- `align-content` (agisce su tutte le righe e le colonne, considerandole come un solo elemento con una sola asse verticale)
    - `flex-start`, `flex-end`, `center`, `stretch` (default), `space-between`, `space-around`, `space-evenly`
        - `stretch`: le linee stretchano per lo spazio rimanente
        - `flex-start`: linee si trovano all'inizio del contenitore
        - `flex-end`: linee si trovano alla fine del contenitore
        - `center`: linee si trovano al centro del contenitore
        - `space-between`: linee sono distribuite equamente nel contenitore
        - `space-around`: linee sono distribuite equamente nel contenitore con spazi intorno a essi
        - `space-evenly`: linee sono distribuite equamente nel contenitore con spazi equali intorno a essi

Ordine (nel flex)

- `order`
    - un numero intero
    - per i singoli oggetti

Allinea oggetto singolo

- `align-self`
    - `auto`, `flex-start`, `flex-end`, `center`, `baseline`, `stretch`

Gap tra gli oggetti

- `gap`
    - due valori, primo per gap tra le righe, secondo gap tra le colonne, se omesso uguale a righe
    - abbreviazione di `row-gap` e `column-gap`

Liste

Tipo di lista

- `list-style-type`
    - `none`, `disc` (default), `circle`, `square`, `decimal`, `decimal-leading-zero`, `lower-alpha`, `upper-alpha`, `lower-latin`, `upper-latin`, `lower-roman`, `upper-roman`, `lower-greek`, `inherit`

- si applica a `ul`, `ol`, `li` (qualsiasi elemento che abbia come display value `list-item`)

Marker posizione

- `list-style-position`
- dove sta il pallino
  - `inside`, `outside` (default), `inherit`

Immagini per marker

- `list-style-image`
  - `url`, `none` (default), `inherit`

Posizione degli elementi

- esistono elementi **block** (tendono a occupare tutto lo spazio a disposizione e vanno a capo) ed elementi **inline** (non vanno a capo)
- questa è una proprietà che dipende dalla proprietà CSS `display`

Posizione

- `position`
  - `static` (default), `relative`, `absolute`, `fixed`, `inherit`
    - `static`: posizione normale
    - `relative`: rispetto al suo posto
    - `absolute`: rispetto al suo contenitore
      - viene usato per i menu dropdown con `display: none;` e `:hover` che cambia il `display`
    - `fixed`: rispetto al viewport
  - `top`, `right`, `bottom`, `left` altre proprietà che "accompagnano" `position`
    - misure in px, em, percentuale, `auto` (default), `inherit`
    - distanza rispetto all'elemento o blocco contenitore
      - **blocco contenitore**: il primo antenato con position non static (o il body), di solito si usa `position: relative;`

Profondità

- `z-index`
  - un numero, `auto` (default), `inherit`
  - il più alto vince

Float

- `float`
  - `left`, `right`, `none` (default), `inherit`
  - sposta l'elemento tutto a destra o sinistra permettendo agli altri elementi di circondarlo, "ignorando" il normale flusso statico
  - si staccano dal flusso normale ma influenzano il contenuto dei blocchi intorno
  - sono contenuti nell'area del contenuto dell'elemento che li contiene
  - i margini sono mantenuti
  - `clear`
    - `none` (default), `left`, `right`, `both`
    - dice se un elemento deve essere spostato sotto e non circondare un float (tipo se metto `left` l'elemento deve stare sotto un float che sta a sinistra)
  - si può usare `overflow` sul contenutore se sfora

Layout Delle Pagine

- Come si struttura la pagina

Fluid

- Proporzionale alla larghezza del browser
- Utilizza `flex`, con un wrapper dove viene indicata la proprietà
- Si ragiona da sinistra verso destra e con le percentuali
- Problema: il testo tende a diventare "stretto" e "verticale" quando si restringe la finestra del browser, oppure le righe diventano troppo lunghe se monitor grande
- problemi con browser piccoli

Fixed

- Self explanatory, indipendente dalla finestra
- si usano i pixel al posto delle percentuali
- anche qui `flex` in un wrapper
- facile, numero di righe controllabile, e di visualizzazione comune per i desktop
- problemi: telefoni del cacchio, rischio di parti oscurate, caratteri grandi = righe corte (in dispositivi con schermo piccolo)

Responsive Web Design

- layout diversi per schermi diversi

- stesso html ma css variabile grazie alle media query

1. Controllare il viewport `<meta name="viewport" content="width=device-width, initial-scale=1.0">`
    - finestra virtuale in cui viene disegnato il sito
    - chiede la misura vera del dispositivo (non quella finta che i dispositivi danno)
    - `user-scalable`: no, yes
    - `minimum` e `maximum-scale`: 1 non scala
2. Gestire layout con media queries `@media (max-width: 768) { .sidebar { display:none;}}`
    - funzionano come degli if
    - `@media [not|only] mediatype and (media feature) { CSS-Code; }`
    - si può usare `and` per utilizzare più media feature
    - è possibile anche metterlo nell'elemento `link` dentro all'attributo `media`
    - Media Types
        - `screen`: versione di default
        - `print`: versione di stampa
    - Media Features
        - `orientation`
        - `max-width`
        - `color`
        - ...
        - si utilizzano molto `orientation` e tutte quelle inerenti alle dimensioni dello schermo
3. usare dei media "fluidi" o "flessibili" `img { max-width: 100%; height: auto;}`
    - anche altri elementi possono utilizzare `max-width` (immagini, video, object, e altri elementi)
    - oppure `background-size` con `contain` e `cover` per le immagini (contain è responsive anche se taglia l'immagine)
    - oppure addirittura mettere un'immagine differente per diverse dimensioni di schermo con un `@media`
    - oppure specificare direttamente immagini diverse a seconda della MQ utilizzando `<source>`
        - `<picture> <source media="(min-width: 650px)" srcset="img_pink_flowers.jpg"> <img src="img_orange_flowers.jpg" alt="Flowers" style="width: auto;"> </picture>`
        - `<source>` specifica una o più risorse media per i tag `<video>`, `<audio>`, e `<picture>`

Breakpoints

- Punti che dividono i dispositivi (piccoli, medi, grandi, ...)

    - `(min-width: ...) and (max-width: ...)`
- Strategie per MQ
    - **Exclusive**: ogni range di larghezza usa regole
    - **Override**: parto da un range di regole e poi le riscrivo per glii altri range
        - *Mobile First* se si parte dagli schermi piccoli (`min-width`)
        - *Desktop First* se si parte dagli schermi grandi (`max-width`)

CSS Variables

- è possibile creare variabili per CSS per evitare di scrivere più volte la stessa cosa e per modificare più facilmente lo stile

Custom Properties

- `:root { --blue: #1e90ff; --white: #ffffff}`
    - `root`: Variabili Globali (radice dell'albero DOM)
    - è possibile anche in classi e id, poi li posso usare nei suoi descendant
    - è possibile riscriverle e sovrascriverle (ad esempio in una media query)
- `var(--blue)`
    - `var(<custom-property-name>, <declaration-value>?)`
    - `declaration-value` è il caso di default
- `--` è obbligatorio

Grid Layout e Bootstrap

Grid

- Griglie verticali che si usano per "armonizzare" la pagina web assegnando diversi ammontare di colonne a ogni elemento
- Parti:
    - Columns
    - Gutter: spazio tra le colonne
    - Row
    - Container (big contenitore)
- Bi-dimensionale
    - righe e colonne
    - Flex era mono-dimensionale

- Grid Lines, definiscono il framework e dividono in celle
  - Numerate per righe e per colonne
  - è possibile dividere in maniera fissa, percentuale, automatica, frazionale etc...

Grid Template

- dire le dimensione delle colonne e delle righe che vogliamo che esistano
- le proprietà vanno applicate a una classe "wrapper" che contiene gli elementi che vogliamo mettere in griglia
- `grid-template-columns`
- `grid-template-rows`
  - soliti modi per definire lunghezze (pixel, auto, percentuali) inoltre:
    - `numerofr`: una unità "frazione" (divide in automatico facilmente)
    - `repeat(numero, lunghezza)`: ripeti una lunghezza un numero di volte
    - `minmax(min, max)`: da una lunghezza minima a una massima, è possibile usare unità differenti

Posizionamento

- `grid-column-start` e `grid-column-end`
  - dicono le colonne in cui un elemento inizia e finisce
  - e.g. `grid-column-start: 1; grid-column-end: 4` l'elemento occuperà le prime 3 colonne (considerare una di più perché si parte da 1)
  - è possibile usare una versione abbreviata: `grid-column: 1/4`
  - se si usa `-1` come valore si conta dalla fine.
    - `1/-1`: tutte le colonne
- `grid-row-start` e `grid-row-end`
  - stesse cose delle proprietà sopra

Grid Template Areas

- è possibile assegnare le "celle" agli elementi in un modo grafico
- `grid-template-areas`
  - i valori sono tante stringhe quante sono le righe con tanti "identificatori" (caratteri o stringhe) quante sono le colonne
  - è possibile assegnare a ognuno degli elementi nel wrapper un identificatore con `grid-area`
  - e.g. `grid-template-areas: "h h h h" "m c c c" "f f f f";`
    `header {grid-area: h;}`
    `nav {grid-area: m;}`

`main {grid-area: c;}`

`footer {grid-area: f;}`

Framework Responsive

- Non dimensioni fisse
- Utilizza dei container variabili con prefissi di classe
- dato che è un framework ci fornisce delle classi e dei seletori da usare
- **W3.CSS**: sistema a frazioni

Bootstrap

- usa il concetto di span (diviso in 12 span), basato su `grid`
- classe base `container` e `container-fluid` (per layout fluido)
- classe `row`
- classi `col` seguito da `-misura` e `-misura` (misura in numero di span/colonne su 12)
  - e.g. `col-sm-4`
  - se si applicano più classi `col` per le differenti misure si ottiene un layout risponsivo
    - `class="col-sm-3 col-md-6"`
  - Misure:
    - `xs`, `sm`, `md`, `lg`
- dimensioni gutter 30px (15px su ogni lato)
- dimensioni decise in precedenza

reset.css

- un file che cambia tutti gli elementi della pagina a uno stato **base**
- utile per partire da zero, al posto di partire da dei framework