Laboratorio 2 - CSS

Appunti	Fatti bene
Comprensione	Quasi tutto
Giorni	26/10

Stylesheet

Inserire nell’HTML il foglio di stile, sotto description:

```
<link rel="stylesheet" href="style.css">
```

Se qui vado a definire media=”screen”, la stampa non sarà influenzata da questo, se non scrivo nulla (valore credo sia “null?” di default) la stampa non viene influenzata da questo.

Al momento non lo specifichiamo perché non ci interessa.

Prime cose che si fanno nel CSS

1. Impostare margini e padding tutti a zero, così evito problemi con le impostazioni predefinite del browser.

```
* {
    margin: 0;
    padding: 0;
}
```

2. Poi definisco le variabili globali per layout:

- con queste variabili definiamo: colore testo, sfondi, link, ecc.
- In generale tutti i colori che compaiono più volte:
  - colore sfondo e testo di header e footer
  - testo e colore background (testo nero, sfondo bianco)
  - colore link e link visitati
  - Colore bandiera del tour di quest’anno (arancione)
  - Un rosso ricorrente
- Definisco anche il font e la famiglia di font (nel caso il mio font non sia disponibile)

```
:root{
    --headerBgColor: rgb(0,0,0);
    --headerTxtColor: rgb(255,255,255);
    --bgColor: rgb(255,255,255);
    --txtColor: rgb(0,0,0);
    --footerBgColor: rgb(0,0,0);
    --footerTxtColor: rgb(255,255,255);
    --linkColor: rgb(230,12,83);
    --visitedLinkColor: rgb(255,255,255);
    --flagColor: rgb(218,103,66);
    --font-body-family: "Roboto Condensed", sans-serif;
}
```

- Posso cambiarli facilmente lo stile dei vari elementi (es. l’anno prossimo cambia il colore del tour, lo modifico da questo file CSS e tutto il sito cambia colore)

Contrasto

- Deve esserci il contrasto di colore tra un serie di elementi:
  - tra testo e background
  - tra link e background
  - tra visitati e non visitate (importantissimo, regola dell’accessibilità, se non rispettata non rispettiamo i requisiti di legge)
  - tra link e colore del testo (se non ce l’ho mi faccio aiutare dalla sottolineatura)

Background

- Definiamo i background così vediamo le aree occupati, poi li posizioniamo con gli ingombri.
- Definiamo colori di background e colori di testo di header, main e footer.

```
header{
    background-color: var(--headerBgColor);
    color: var(--headerTxtColore);
}

main{
    background-color: var(--bgColor);
    color: var(--txtColor);
}

footer{
    background-color: var(--footerBgColor);
    color: var(--footerTxtColor);
}
```

Nav

- Duplichiamo per il menù (nav)

```
nav{
    background-color: var(--headerBgColor);
    color: var(--headerTxtColore);
}
```

Header

- Centriamo l'immagine, per ora la consideriamo come testo, poi dopo faremo image replacemente col CSS.

```
header{
    background-color: var(--headerBgColor);
    color: var(--headerTxtColor);
    text-align: center;
    padding: 1em; /* Messo io */
}
```

HTML

Di solito metto nell'html tutte le caratteristiche che devono valere per tutti gli elementi, che poi potrò modificare da qui.

- `font-size: 100%;` → rispetta esattamente quello che vuole l'utente. Questa sarà la dimensione del font sul paragrafo, per gli 'h' la aumenteremo dopo.

```
html{
    font-size: 100%;
    font-family: var(--font-body-family);
    line-height: 1.5em;
    margin: auto;
}
```

- Definisco la font family
- Per l'accessibilità: interlinea di almeno 1,5em
- Per centrare elemento `margin: auto;`
  - Mentre per centrare testo: `text-align: center;`

Background

- L’immagine di background posso darla a `main` e ad `html`. In questo caso lo do nell’html perché, non tanto nell’index, ma nelle altre pagine uso `main` per centrarlo.
- Il background copre i padding ma non i margini
- Inserisco l’url dell’immagine
- Se l’immagine non fosse chiara (ma scura), dovrei definire un background color scuro. Perché altrimenti di default sarebbe bianco, e se l’immagine non arrivasse o ci fossero altri problemi, avrei scritte chiari su sfondo chiaro.
    - In questo caso è giusto non definirlo perché stiamo inserendo un’immagine chiara, in tutti gli altri casi va definito un background scuro.

  Inseriamo l’immagine di back ground:

```
html{
    font-size: 100%;
    font-family: var(--font-body-family);
    line-height: 1.5em;
    margin:auto;
    background-image: url("LabHTML/materialePerStudenti/images/bg.png");
}
```

Menù

- Differenziare tra nav con id=”breadcrumb” e nav con id=”menu”

```
#menu{

}
#breadcrumb{

}
```

Esercizio per lo scritto:

- Se io definisco

```
#menu{
    width: 40%
}
```

- Il menù sarà largo 50% perché la seconda ha più specificità di questa (non so quale fosse la seconda ma avrà avuto un tag HTML)
- La prima regola ha specificità 1, 0, 0, la seconda 1, 0, 1, perciò viene applicata la seconda.

Elementi dentro lista

```
#menu ul li{
    display: inline-block;
}
```

- Gli ‘li’ erano già elementi di blocco, se gli do `display: inline` diventano elementi di linea.
- Invece se gli do `display: inline-block;` rimangono elementi di block, perciò, ad esempio, posso dare un padding, ma non vanno a capo. Ora ho una lista orizzontale.

Link e link visitati

```
#menu a:link{
    color:var(--LinkColor)
}

#menu a:visited{
    color:var(--visitedLinkColor)
}
```

Menu ul

Devo centrare la lista “ul”

```
#menu ul{
    margin:auto;
```

```
}
```

- Il menù non termina dove termina il suo contenuto, ma occupa tutta la linea perché è inline.
- Se gli do solo `margin: auto;` prende la differenza tra l'ingombro e il padre e lo divide per 2. Ma in questo caso fa 0 diviso 2 che rimane 0 e quindi non lo centro
- Perciò dobbiamo andare a dargli una dimensione in lasrghezza massima (non una larghezza fissa) così facciamo un layout responsive.

- Se do una larghezza fissa, es. 800px, il problema è che non centra più il menù e se riduco la dimensioni della pagina compare lo scroll sotto.

```
#menu ul{
    margin:auto;
    width:800px;
}
```

- Perciò è molto meglio dargli una larghezza massima:

```
#menu ul{
    margin:auto;
    max-width:1024px;
}
```

- Dandogli una larghezza massima possiamo stare dentro i 1024 e mi permette di restringere la pagina quanto voglio.
- Quando usiamo larghezza massima posso usare i pixel.
- Se dobbiamo definire una larghezza, per avere un layout flessibile, devo usare la %.

Padding

- Diamo del padding agli elementi in modo che si possano distribuire.

```
#menu ul{
    max-width:1024px;
    padding: 1em;
    margin:auto;
}
```

```
#menu ul li{
    display: inline-block;
    width: 20%;
    margin: 0 0.5em;
}
```

- Se divido lo spazio in 4 parti, dovrei scegliere 25%, ma sto un più sotto (20%) perché così riesco a dare dei margini
- margine sopra e sotto 0, sinistra destra 0,5em (posso scegliere margin o padding, è lo stesso)

- Dobbiamo togliere i punti dell'elenco puntato:

```
#menu ul{
    max-width:1024px;
    padding: 1em;
    margin:auto;
    list-style-type: none;
}
```

- Scriviamo lo stile dell'header una volta e poi linko lo stesso per tutte le pagine. Faccio un file unico per tutti. Se si fanno più file statisticamente si ha un incremento del consumo dei byte maggiore del 35%.
- Quando nel sito si hanno diverse immagini piccole, è meglio inserirle tutte in un'unica immagine che viene richiesta e mandata all'utente con un unico pacchetto, piuttosto che chiedere diversi pacchetti al server.
    - Poi, per fare il crop delle varie immagini piccole si usa il tag area
- Spedire un pacchetto da 1MB è più veloce rispetto a 10 pacchetti da 100KB, perciò si tende a preferire l'invio di pacchetti grandi.

- Ha messo il text align dentro il menu generico, è una scelta stilistica di programmazione

```
#menu{
    text-align: center;
}
```

- Andiamo a definire dentro nav, altrimenti dobbiamo rifarlo

```
nav a:link{
    color: var(--linkColor);
}

nav a:visited{
    color: var(--visitedLinkColor);
}
```

- Se lo definiamo da entrambe le parti nel caso volessimo fare cambiamenti in futuro, dovremmo fare così per tutti gli ID.
- La prof tende a definire la proprietà “centrato” nel nodo dell’albero della pagina più in alto per evitare le ripetizioni, così in caso di modifiche lo modifica una volta sola.
    - Però l’altro modo non è sbagliato, dipende dal fatto se in futuro andremo a fare modifiche

---

Paragrafi del breadcrumb

```
breadcrumb p{
    max-width: 1024px;
    margin: auto;
}
```

---

3 colonne immagine e testo

```
#imgIndex{
    width: 80%;
}
```

Larghezza dell’immagine rispetto alla colonna

- Abbiamo una lista, che contiene dei div, che contiene 1 dt e 2 dd
- La lista è contenuta in un main in due versioni: una con le 3 colonne e una con la tabella.

Creiamo la classe colonne:

- Non gli diamo id perché quando è stato definito html5 si sono dimenticati di un pezzo. Prima potevo creare un div con un id=”main” e quell’id main potevo usarlo per creare link al tag main. Adesso che ho un **tag** main non c’è più un modo di creare un link a quel tag, si sono dimenticati di questa necessità.
    - Posso usare gli id su css, javascript e come destinazione per i link
    - Una cosa importante è dare la possibilità di saltare il menù, cioè creare un link che vada direttamente al contenuto (utile per gli screen reader: altrimenti verrebbe riletto il menù in alto, sempre uguale, per ogni nostra pagina).
      I “torna su” funzionano allo stesso modo: hanno un link ad un ‘#’ della pagina oppure al main.
    - Adesso siamo in una situazione in cui tutti i main hanno un id come “content” o ”main” perché non c’è possibilità di accedervi come tag.
- Creiamo id=”colonne” e id=”tabelle” per la due diverse versioni della pagina. Lei le lascia sotto un unico id=”content”, così poi non dovrà andare a modificare i link quando li inserirà (?).
- Creiamo per questa pagina la classe colonne, quando faremo la tabella faremo un’altra cosa.
- In questa pagina particolare, con le colonne, avremo delle regole per questo main, nei main delle altre pagine farò altre cose (es. la biografia sarà più stretta).

```
main.colonne{
    column-count: 3;
    width: 90%;
}
```

- Spesso si inserisce l'immagine nel body dell'html e la larghezza massima del body nel CSS.

```
main{
    color: var(--txtColor);
    margin: auto;
    width: 80%;
}
```

- Così do a tutti i main un `margin: auto;` e le altre 2 caratteristiche.

- Mentre per il main che avrà la classe colonne faremo una cosa diversa:

```
.colonne dl{
    font-size: 90%;
    margin: auto;
}
```

- Prima cosa da fare: dare a questa lista la proprietà di essere in-line come nell'altro caso (quale?)
- Abbiamo una lista di definizioni qui e una nella lista dei componenti.
- Vado a definire una regola molto specifica in modo che quello che applico qua non vada sui componenti. Farebbe confusione (?)
- I dt sono dentro ai div, che sono all'interno di un main che ha classe colonne.
  - La specificità è: 0 1 3
    - 0 id
    - 1 attributi: la classe ".colonne"
    - 3 tag html: main, div, dt

```
main.colonne div dt, dd{
    display: inline;
}
```

⚠️ Scritto così, quello che scrivo sul dt vale anche per il dd? NO, errore comune!!

Devo riscrivere tutto `main.colonne div dt, main.colonne div dd{` così:

```
main.colonne div dt, main.colonne div dd{
    display: inline;
}
```

es. Se ci chiedesse la specificità di queste regola: dovremmo dire che in realtà sono due e dovremmo calcolarle separatamente (dt e dd). In questo caso le indichiamo come uguali.

Diamo la classe colonne al main

```
<main class="colonne">
```

//tolto nel main il background color, credo perché se lo sfondo è chiaro non serve specificarlo

---

Lettere maiuscole

```
main.colonne div dt, main.colonne div dd{
    display: inline;
    text-transform: uppercase;
}
```

Stile Citta

- Più diamo specificità più siamo sicuri che quello stile di visualizzazione non viene applicato da altre parti.

  ```
  dd.citta{
      color:var(--flagColor);
      font-weight: bold;
  }
  ```

- //lei ha scritto qualcosa come `dd[data-title="Citta"]` al posto di `dd.citta` (ha detto che va a prendere solo i dd che hanno data-type citta)

- Dà alla città il colore della bandiera e la rende in grassetto

---

Stile Data

- Metto in grassetto il numero del giorno della data (contenuto nello span, che è dentro al tag time, dentro al div).
- Mettere questo span nell'HTML è un po' una rottura della separazione presentazione/struttura; ma, visto che lo span non ha nessuna connotazione semantica, non è un gran problema e al momento non c'è altro modo per farlo.

```
main.colonne time span{
    font-weight: bold;
}
```

Padding main

```
main dl div{
    padding-top: 0.3em;
    padding-bottom: 0.3em;
}
```

- Come decido se 0.3 o 0.7 di padding? Provo con gli strumenti per sviluppatori e vedo come viene.

Colonne vere e proprie

Aggiungo nel main la class colonne

```
<main class="colonne">
```

Definisco le colonne

```
main.colonne{
    column-count: 3;
```

```
    width: 90%;
}
```

//footer va messo dentro al body, ma fuori dal main

Sistemiamo il footer

- Facciamo “una magia” non del tutto corretta, aggiungiamo il padding. Perché se metto un contenuto più corto (che non riempie la pagina) il footer mi sale. Invece lo vorrei sempre attaccato al fondo.

```
footer{
    background-color: var(--footerBgColor);
    color: var(--footerTxtColor);
    padding: 1.0em;
    text-align:center;
    margin-top: auto;
}
```

- Se il contenuto è più piccolo (è breve, non occupa tutto lo spazio) della pagina, vediamo come mettere il footer sul fondo della pagina domani in aula.
- Copiamo il link di stile anche nella pagina dei componenti (e nelle altre).

Consiglio

- Ricordiamoci di copiare il sito sul validatore del w3c per vedere se il nostro codice è valido (come quando nella programmazione compiliamo regolarmente il codice).

Manca e facciamo domani

- sold out rovesciato
- img pinguini tattici nucleari nel titolo

Prossimo lab

- prossimo lab: vedremo componenti e tabella

Lab quello dopo ancora: smartphone e una cosa che non ho capito

---

Per casa

- validare quello che abbiamo fatto
- Proviamo a fare la biografia (non la faremo assieme). Se troviamo problemi nel prossimo lab ne parliamo