**LEZIONE 2**

<https://modern-coast-d4b.notion.site/Primi-concetti-e-TAG-ff4a90f318fd40cd9f55e61f76f2d665>

**LEZIONE 3**

<https://modern-coast-d4b.notion.site/Analisi-pagine-web-5a716752b4bc43d3b7d9b419cc3df428>

**LEZIONE 4**

<https://modern-coast-d4b.notion.site/Container-con-cards-351bf0ab07b94e5a9a8b5b0ea506678b>

**LEZIONE 5**

<https://modern-coast-d4b.notion.site/Header-Nav-Background-c24d731d8543439dbb26bce1085de1af>

**LEZIONE 6**

<https://modern-coast-d4b.notion.site/Navbar-e-form-32ddea6e835a4dd192dc66550f4048fa>

1 LEZIONE

importanza di avere **un profilo digitale**

importanza nel saper scrivere con **la tastiera** senza guardare, conoscere il **terminale**, e il proprio **editor**

fondamentale conoscere **le shortcut** del proprio editor

obbligatorio testare i propri codici su **TUTTI i browser**  (chrome, firefox, safari, browser android)

fondamentale **conoscere l’inglese** per capire nel dettaglio le varie regole di base di programmazione, la documentazione è realizzata e aggiornata principalmente in INGLESE.

**README** è un file di testo che contiene delle **informazioni importanti** per: l’utilizzatore, per lo sviluppatore, ai colleghi che magari non conoscono il codice da noi creato (includono link ed informazioni anche esterne)

INTRODUZIONE AL LINGUAGGIO DI PROGRAMM. HTML CSS E JS

* **HTML** 🡪 indica la struttura e che può ospitare il contenuto di un sito
* **CSS** 🡪 indica la parte di stile utilizzato per definire l'aspetto e la formattazione di documenti HTML
* **JavaScript** 🡪 indica il funzionamento e la **logica**, per le operazioni dinamiche sul nostro sito

-**HTML Boilerplate** 🡪 è un modello di codice HTML di base che **fornisce una struttura iniziale** per una pagina web.

include gli elementi fondamentali necessari per la struttura di una pagina web, come ad esempio la dichiarazione del tipo di documento, l'elemento HTML radice, l'elemento head e l'elemento body.

-**<!doctype html>**🡪indica il tipo del documento, la versione del linguaggio che stiamo utilizzando (**HTML:5**), serve al browser perchè deve sapere come interpretarlo, quindi deve essere in cima al codice.

nel caso venisse tolto, i vari browser possono leggere il nostro codice in modo diverso.

-**HTML element** 🡪 serve come contenitore che contiene gli elementi di tutta la struttura, mettere elementi al di fuori dell’elemento HTML è un errore!!, non sbagliare per non mostrare la UIX all’utente con degli errori e spazi vuoti.

-**<html lang = en>** 🡪 è un **attributo** (ha delle specifiche in + di un elemento normale), è utilizzata per specificare **la lingua principale** del contenuto. Inoltre fornisce informazioni al browser e ad altri agenti utente (collaboratori) sulla lingua predominante.

-**HTML head** 🡪 viene utilizzato per includere informazioni relative al documento HTML, come **metadati**, collegamenti a **fogli di stile** (CSS), **script JavaScript** (JS) ecc.. può contenere anche altri elementi in base alle esigenze specifiche del tuo sito web.

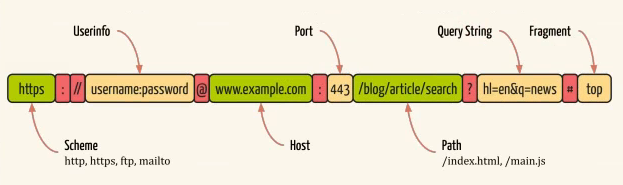
-**meta** 🡪per fornire metadati sulla pagina web. I metadati forniscono delle **informazioni** su un altro dato, come il set di **caratteri utilizzato**, la **descrizione della pagina**…

-**charset="utf-8"** 🡪 indica il set di caratteri utilizzato nel documento, assicura che il browser interpreti correttamente i caratteri presenti nella pagina (c’è una tabella che traduce i bit in una lista di caratteri 8 bit)

-**HTML body** 🡪 viene utilizzato per racchiudere tutto il contenuto visibile quando si visita una pagina web (titoli <h1>, paragrafi <p>, links <href> …)

-**URL** 🡪 (**Uniform Resource Locator**) è un indirizzo che identifica in modo univoco una risorsa su Internet. Gli URL sono comunemente utilizzati per accedere a pagine web:

* **Schema** 🡪 stabiliamo il protocollo di **comunicazione** (ftp, HTTP, HTTP**S**)
* **Userinfo** 🡪 le credenziali dell’utente che vengono richiesti in alcuni protocolli
* **:** **@**  **// #** 🡪 sono identificati come TOKEN
* **Host** 🡪è la macchina con cui vogliamo comunicare, viene letto l’indirizzo (identifica univocamente una macchina nella rete 256^4) dal **servizio DNS** (associano l’indirizzo IP ad un nome)
* **Port** 🡪 è collegata alla scheda di rete, essa ha diverse porte e deve identificare quella corretta perché possono esserci diversi servizi
* **Path** 🡪 è il percorso che specifica la posizione di una risorsa su un server web che è richiesta, NON sono solo file e cartelle (index.html, main.js
* **Query string** 🡪 sono dei parametri passati al server
* **Fragmant** 🡪è utilizzato per identificare una sezione specifica di una risorsa web



-**COOKIE** 🡪 sono delle stringhe di testi salvati un certo formato, che si usano per memorizzare dei dati di qualcosa, “***spiano***” gli interessi e comportamenti delle persone per fare del marketing mirato. Importante dire che i COOKIE *hanno una scadenza.*

-**PARSING** 🡪 l’input viene letto un carattere alla volta per interpretare quello che c’è scritto, è un processo di analisi e interpretazione del codice HTML da parte di un browser o di un altro programma, è essenziale per garantire che le pagine web vengano interpretate e visualizzate correttamente dai browser.

-**IL PARAMETRO ?v=1.0** 🡪 viene utilizzato per indicare la versione di una risorsa, come un file CSS o JavaScript, questo può essere utile **per gestire la cache del browser** e garantire che i visitatori visualizzino la versione più recente di un file quando viene aperta la pagina.



-**La cache del browser** 🡪 è un **database** di file utilizzato per memorizzare le risorse scaricate dai siti web, tra le risorse più comunemente memorizzate nella cache di un browser sono:

immagini, video, contenuto testuale, HTML, CSS e Javascript.

è una ***parte essenziale*** del modo in cui i browser web gestiscono e accelerano il caricamento delle pagine web e migliorano le prestazioni durante la navigazione, parlando del suo funzionamento:

* **Richiesta iniziale della pagina**: visitiamo una pagina web per la prima volta, il tuo browser invia una richiesta al server web per ottenere tutti i dati necessari per visualizzare la pagina (HTML, CSS, JavaScript, immagini…)
* **Memorizzazione nella cache**: Una volta che il browser ha ottenuto i dati, salva copie di essi nella memoria cache locale del tuo computer. La cache è una sorta di **archivio temporaneo** che consente al browser di **accedere rapidamente ai file necessari senza doverli scaricare nuovamente** dal server ogni volta che visiti la stessa pagina o altre pagine che utilizzano gli stessi elementi.
* **Controllo di validità**: la cache deve essere gestita per assicurarsi che i dati siano sempre aggiornati. il browser deve *verificare periodicamente* se i file nella cache sono ancora validi o se sono state apportate modifiche; viene messo in pratica attraverso le informazioni di controllo della cache nel protocollo HTTP.
* **Eliminazione della cache**: A volte, è necessario eliminare la cache del browser, ad esempio, quando si verificano problemi di visualizzazione delle pagine. questo può essere fatto manualmente tramite le impostazioni del browser.

-**scheda di rete** 🡪è un **componente hardware** che consente a un dispositivo, come un computer o un server, di connettersi a una rete e comunicare con altri dispositivi sulla stessa rete o su Internet. La scheda di rete gestisce la trasmissione e la ricezione dei dati attraverso il collegamento di rete.

-**sistema operativo (OS)** 🡪 è un **software che gestisce le risorse hardware** e fornisce un'interfaccia tra l'utente e il computer. Esso facilita l'esecuzione di programmi, la gestione dei file, la comunicazione con l'hardware e fornisce funzionalità di base come la sicurezza e la gestione delle risorse.

-**COMPABILITA** 🡪cercar di far funzionare il mio codice funziona **su differenti browser** (dai più vecchi ai più recenti), cercando di rimuovere versioni di browser e browser stessi (internet explorer morto 2 anni fa), per ridurre al minimo i problemi.

-**Server DNS** 🡪 riceve il www.Xsito, lo **converte in un indirizzo IP** che permette all’utente di visitare la pagina, la risorsa viene ricercata con una determinata gerarchia d’importanza (index.html, file css), l’ultimo pacchetto contiene un informazione che la trasmissione di comunicazione è terminata.

se non viene trovata nessuna risorsa da restituire, si restituisce un ERROR 404 (**risorsa non trovata**)

-**pacchetti** 🡪 insieme di dati con ***valore* 0 1**, questi pacchetti vengono trasmessi tramite la scheda di rete, verso il router e successivamente attraverso la rete Internet raggiungono il server di destinazione.

-**separatori / TOKEN** 🡪 i separatori sono gli spazi vuoti tra caratteri, i token sono le sequenze

-**TAG PLACEMENT** 🡪 quando la pagina incontra il collegamento script a **JS** … il browser legge **src** che sta per source ed **è un attributo**), la risorsa mi indica **l’URL**( anche se non è come l’URL che troviamo nell’URL del browser in quanto non abbiamo ex un protocollo) del file **JS** che verrà succesivamente scaricato, ma prima di proseguire con la struttura di HTML, deve svolgere il **procedimento dei pacchetti, scheda di rete, parsing, e separato per token ecc…**

Quindi, **è meglio metterlo prima in cima nella parte head, oppure nella porte body**? 🡪 meglio metterlo **DOPO** in quanto almeno si carica e appare qualcosa sulla UX interface dell’utente, anche se la pagina non è ancora elaborata completamente

(importante i primi 3 secondi!!)

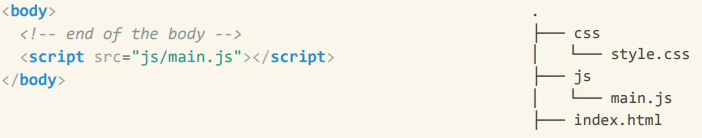


COLLEGARE UN FILE CSS A HTML

è molto più LOGICO, SINTETICO e ORDINATO distinguere i vari file di linguaggio diverso in locazioni diverse,posssono essere **INLINE** oppure **EXTERNAL** o anche **EMBEDDED** , per collegare i file **CSS** al **HTML:**



**STRUTTURA** delle cartelle e file:



Esistono 2 tipi di “posizioni” di dove si trovano i file:

* **REMOTO** **:** quando il file da collegare NON è sulla stessa macchina
* **LOCALE:**  quando il file da collegare è sulla stessa macchina



2 LEZIONE

STORIA DI HTML

HTML è sato creato da **Tim Berners-Lee**, che pubblico la sua prima pagina il 6 agosto 1991, facciamo una lista delle varie versioni di HTML rilasciate nella storia:

* HyperText Markup Language
* Early 90s
* HTML4 in 1997
* XHTML nel 2000 🡪 un file XHTML ha tag elementi, ma a differenza di html:5, ha la caratteristica che I nomi dei tag erano a piacere, è obbligatorio il tag di chiusura
* HTML5 nel 2014

TERMINOLOGIE HTML

-**web design** 🡪 è il processo di pianificare, strutturare e creare un sito web

-**web development** 🡪 il processo di programmare applicazioni web dinamiche

-**front end** 🡪 si concentra sull'interfaccia utente e sull'esperienza dell'utente, la parte dell'applicazione con cui gli **utenti interagiscono** direttamente. È l'interfaccia utente che vedono e con la quale interagiscono attraverso il browser o l'applicazione stessa

-**backend** 🡪 è il funzionamento interno e la funzionalità di un sito web o di un’applicazione, gestisce l'elaborazione dei dati e la logica di business. È responsabile delle operazioni che avvengono dietro le quinte (es: l'elaborazione dei dati)

INTRODUZIONE SULL’ANATOMIA DI HTML

Prima di addentrarci nei vari elementi in HTML, deifiniamo prima l’anatomia di un elemento HTML:

* **ELEMENTO** 🡪 sono elementi costitutivi delle pagine web, un singolo componente dell’HTML, come esempi abbiamo: paragrafo, tabella, elenco, div, ancoraggio …



* **TAG** 🡪sono dei “marchi” di apertura all’inizio di un elemento, ma anche dei marchi di chiusura alla dine di un elemento. i TAG contengono dei caratteri che indicano il significato del TAG



* **ATTRIBUTO** 🡪 fornisce delle informazioni aggiuntive sull’elemento HTML, è posizionato all’interno di un TAG DI APERTURA, esempi di attributi: class, id, style…



* **VALORE (VALUE)** 🡪 è il valore assegnato che viene usato dall’attributo

INTRODUZIONE AGLI ELEMENTI DI HTML

-**DOCTYPE** 🡪 è il primo elemento presente in una pagina HTML, che dice al browser:

**quale versione del linguaggio di markup la pagina sta utilizzando**, il formato del DOCTYPE si è modificato di versione in versione di HTML oggi è ( **<!doctype html>** ).

-**HTML TAG** 🡪 si trova dopo il DOCTYPE, il contenuto di tutta la pagina deve essere compresa tra gli <html> tags.

-**HEAD** 🡪 contiene il **titolo della pagina** e i **meta dati** sulla pagina (ricordiamo che sono informazioni non sono visibili all'utente, ma hanno molti scopi, ad esempio: fornire informazioni ai motori di ricerca e ai vari collaboratori che visualizzano il codice).

-**BODY** 🡪 contiene tutto il contenuto che è **VISIBILE** dall’utente nel contenuto della pagina

-**COMMENTI** 🡪 i commenti scritti all’interno del codice (anche se nel body) non verrano mai visualizzati dall’utente, ma solo dalla visuale dell’editor, sono molto utili per tenere ordine e spiegare a chi non conosce il codice cosa si sta facendo



-**PARAGRAFI <p>** 🡪 ci permettono di creare del testo e di formattarlo in maniera elegente

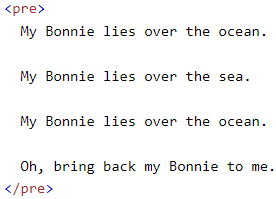
-**TITOLI <h1, h2…>** 🡪 vanno a sclare e vengono ***usati come gerarchia in base all’importanza*** del titolo (dal + importante h1, ai – importanti h6)

-**FORMATTAZIONE DEL TESTO** 🡪indichiamo tramite alcuni tag l’importanza di indicare un significato ad esempio in un paragrafo, tra i + noti abbiamo:

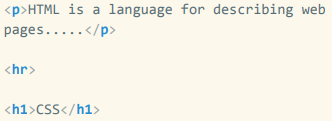
* **Em** 🡪 trasforma quel pezzo in corsivo
* **Strong** 🡪 mette quel pezzetto compreso tra i tag in **grassetto**
* **b** 🡪 sta per bold
* **i** 🡪 sta per italic (corsivo)

-**LINE BREAK**  **<br>**🡪 permette di andare a capo rompendo il testo (non è un abitudine buona mettere dei line break all’interno dei paragrafi)

-**<pre>** 🡪 è un’ottima alternativa al <br>, in quanto definisce il testo preformattato. Conserva sia gli **spazi** che le **interruzioni** di una riga:



-**<hr>** 🡪 crea letteralmente una linea, viene usato solitamente per cambiare tematica nel contenuto



-**ANCORA <a>** 🡪 non è link ma si chiama elemento ANCORA esiste solo con l’**href**, il valore nel href all’interno dell’ancora con **# (fragment ed è un URL) ,** ci porta al valore che ha il nome ricercato cliccando il link.

I collegamenti normali puntano sempre alla pagina superiore di una pagina

Le ancora possono puntare a un specifico punto della pagina tramite l’**ID**

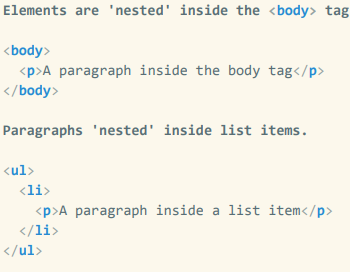
**\_BLANK** 🡪 Apre il documento collegato in una nuova finestra o scheda

**\_SELF** 🡪 Apre il documento collegato nella stessa cornice in cui è stato fatto clic (**impostazione predefinita**)

**\_PARENT** 🡪 Apre il documento collegato **nel frame principale**

**\_TOP** 🡪 Apre il documento collegato nell'intero corpo della finestra

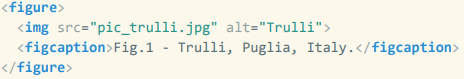
**Framename** 🡪 Apre il documento collegato in una cornice con nome

-**NESTING** 🡪 gli elmeneti sono “incasrtari” l’uno con l’altro dentro il <body> tag

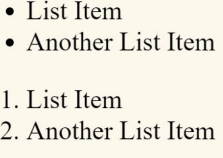
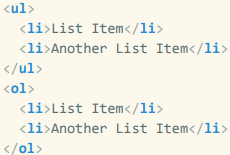
-**IMMAGINE** 🡪 le immagini hanno 3 parti **+ DETTAGLI SU TIPO** 🡪 [CLICCAMI!](../ESERCIZI/02-html-elements/06-images/presenzazione-image.pdf)

* <img *“contenuto”* />
* **attrbiuto src**: cerca di localizzare dove si trova il file dell’immagine
* **alt**: una breve descrizione dell’immagine (utile in caso la pagina non carica l’immagine)

-**FIGURE** 🡪 può contenere e racchiudere l’immagine + la sua didascalia



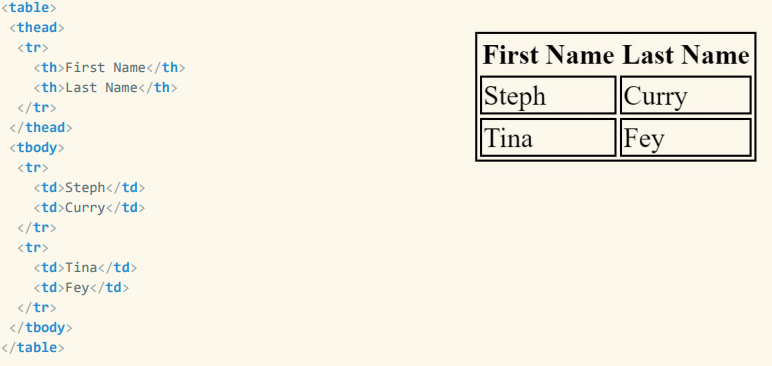
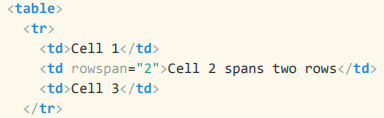
-**LISTA** 🡪 possiamo anche delle liste ORDINATE (ordine numerico **<ol>**) o anche DISORDINATE (pallini **<ul>**), all’interno di questi 2 tag madri, mettiamo **<li>**

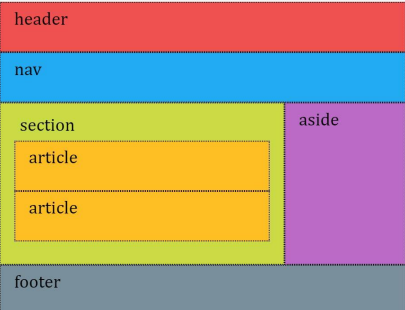
 

-**TABELLE** 🡪 Le tabelle sono un modo per rappresentare informazioni complesse in un **formato griglia**, sono costituite da RIGHE DI INTESTAZIONE (thead) con i **nomi di colonna** e RIGHE DI CORPO con i **dati** (tbody), avviene in ***ORIZZONTALE.***

Bisogna anche menzionare 2 attributi riguardanti le tabelle:

* **Rowspan** 🡪 specifica quante righe si estende **VERTICALMENTE** una cella
* **Colspan** 🡪 specifica quante righe si estende **ORIZONTALMENTE** una cella



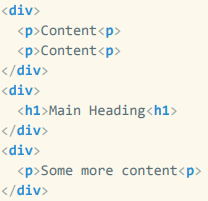
-**HTML5 ELEMENTS** 🡪 la nuova versione attuale di HTML usata, offre un documento migliore con una **struttura e semantica molto valida**, tra i tag SEMANTICI + importanti che utilizziamo per rendere una pagina molto pulita abbiamo:

* <header>
* <nav>
* <section>
* <article>
* <aside>
* <footer>

**Danno un significato** preciso al contenuto

-**DIV** 🡪 il DIV è un elemento a livello di **blocco**:

* Una divisione o sezione di contenuto all'interno di una pagina HTML.
* Utilizzato per raggruppare elementi per formattarli con CSS.
* Applicare **ID** e **classi** ai div per controllarne gli stili con i CSS.



-**SPAN** 🡪 molto utile per lavorare con elementi inline da inserire dentro elementi **block**

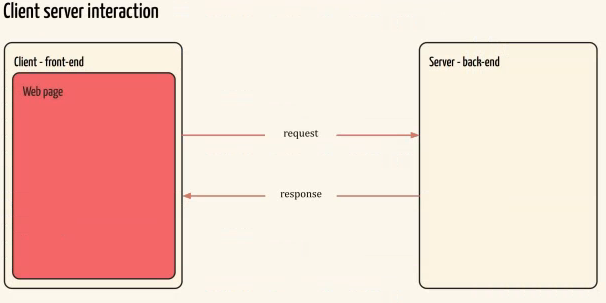


3 LEZIONE (FORM)

Cosa sono i forms? In italiano sono **moduli**, ci permettono di:

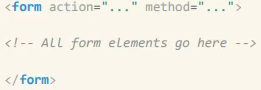
* **Raccogliere dati dall’utente** 🡪 **+ nello specifico**: registrazione e accesso ai siti web ○ inserimento di informazioni personali (nome, indirizzo, dati della carta di credito...), filtraggio dei contenuti (utilizzando menu a discesa), esecuzione di una ricerca ○ caricamento di file
* **Contengono elementi detti controlli** 🡪 Input di testo, caselle di controllo, pulsanti di opzione, pulsanti di invio…
* **Quando gli utenti compilano un modulo** 🡪 i dati vengono ***inviati a un server Web*** (back-end) per l'elaborazione

schema del mandare e ricevere richieste:



-**form** 🡪è un ***attributo*** che specifica il modulo <input> a cui appartiene l’elemento, tutti gli elementi del form vanno dentro il <form> </form>, ha due attributi principali:

* **Action** 🡪specifica l'URL o il percorso del server che gestirà i dati inviati dal form.
* **Method** 🡪 specifica il metodo HTTP utilizzato per inviare i dati al server (GET e POST)



-**form action** 🡪 una volta completati tutti i campi e cliccato il tasto SUBMIT, verremo reindirizzati sulla pagina .php



-**form method** 🡪 I dati dei form possono essere inviati al server utilizzando **2 METODI PRINCIPALI**:

* **get** 🡪 sono visibili nella barra degli indirizzi del browser.

i **vantaggi** sono: I dati inviati dal metodo GET vengono visualizzati nell'URL, è possibile aggiungere ai **segnalibri la pagina** con valori specifici della stringa di query (possono tornare alla stessa risorsa con i dati già compilati)

**Gli svantaggi**: non adatto per il passaggio di informazioni sensibili come nome utente e password, la lunghezza dell'URL è limitata.

* **post** 🡪non sono visibili all'utente.

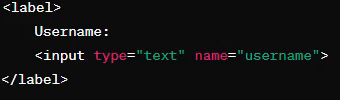
**vantaggi** sono: + sicuro di GET; le informazioni non sono mai visibili nella stringa di query dell'URL o nei log del server, ha un limite molto più ampio sulla quantità di dati che possono essere inviati, può inviare dati di testo e dati binari (caricamento di un file).

**Gli svantaggi**: non è possibile aggiungere la pagina ai segnalibri con la stringa di query

BASIC FORM ELEMENTS

-**label** 🡪 viene utilizzato per fornire una descrizione e un “**collegamento**” per un elemento di input all'interno di un modulo. Questo elemento è utile per migliorare l'accessibilità e l'usabilità del modulo stesso. Vedremo che può essere utilizzato in **combinazione** 🡪 **for = id**

Il **name** serve per il back-end, scriverlo sempre in inglese mai in altre lingue (crea confusione)



-**text field** 🡪 contiene un campo di testo singolo.



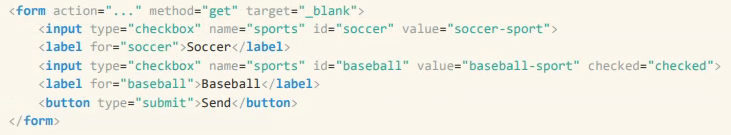
-**password field** 🡪 campo di testo per inserire password.



-**radio buttons** 🡪 pulsante di opzione che permette all'utente di selezionare **UNA SOLA** opzione da un gruppo di opzioni. il valore verrà inviato al back-end quando l'opzione viene selezionata.



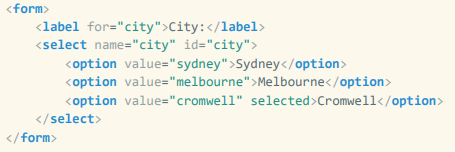
-**checkbox**🡪 Casella di controllo che permette all'utente di selezionare **+ DI UN’** opzione.

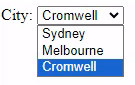




-**drop downs menu** 🡪 offre la possibilità di selezionare più opzioni. Utilizzare <select> i pulsanti di opzione quando il numero di opzioni tra cui scegliere è elevato. Per non dare nessun’opzione iniziale usiamo:

**value="" disabled selected hidden>Seleziona una opzione**



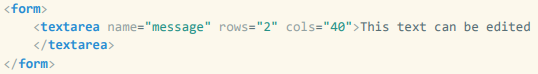


-**file select** 🡪 caricare un file locale come allegato, dopo aver selezionato un file in upload, potremmo visualizzarlo sulla pagina



-**text area** 🡪elemento che permette agli utenti di inserire testo su più righe. Possiamo definire sia:

le rows (righe in altezza) cols (spazio in orizzontale), possiamo dare anche un limite massimo di caratteri inseribili: **maxlength="255"**



-**submit and reset** 🡪 vengono spesso integrate insieme, le differenze sono:

* **submit** 🡪 pulsante che invia i dati del form al server.
* **Reset** 🡪 pulsante che reimposta i valori dei campi del form ai loro valori predefiniti.



-**fieldset** 🡪 viene utilizzato per **raggruppare elementi correlati** all'interno di un modulo e creare una suddivisione visiva o una sezione. È spesso utilizzato insieme all'elemento **<legend>,** che fornisce una didascalia o un titolo per il <fieldset>. HA UN ***SIGNIFICATO SEMANTICO***



-**email field** 🡪 usato per ricevere una email valida dall’utente, molti browser possono verificare la sua validità senza l’utilizzo di JS.



-**number** 🡪 utilizzato quando dobbiamo inserire dei parametri come il peso di una persona.



FORM VALIDATION

La convalida è un meccanismo per garantire **la correttezza dell'input dell'utente**, può essere utilizzata per:

* Assicurarsi che siano state inserite tutte le informazioni richieste
* Limitare le informazioni a determinati tipi (ad es. solo numeri)
* Assicurarsi che le informazioni seguano uno standard (ad es. e-mail)
* Limitare le informazioni a una certa lunghezza tramite esempio **maxlenght**

Per maggiori dettagli 🡪 [Client-side form validation](https://developer.mozilla.org/en-US/docs/Learn/Forms/Form_validation)

La convalida deve essere eseguita sia dal **front-end** che dal **back-end** (sono due parti fondamentali di un'applicazione software o di un sito web.)

-**Front-end** 🡪 L'applicazione deve:

* convalidare tutte le informazioni per assicurarsi che siano complete, prive di errori e conformi alle specifiche **richieste dal back-end.**
* dovrebbe contenere meccanismi per avvisare gli utenti se l'input non è completo o corretto
* dovrebbe evitare di inviare dati "cattivi" al back-end

-**Back-end** 🡪 deve eseguire la propria convalida di tutti i dati ricevuti:

* Non deve MAI fidarsi del fatto che il front-end abbia eseguito la convalida
* Alcuni utenti intelligenti possono bypassare facilmente i meccanismi front-end
* I servizi back-end possono ricevere **dati da altri servizi**, non necessariamente front-end

4/5 LEZIONE

INTRODUZIONE AL CSS 🡪 Cascating Style Sheets, è uno style sheet language

Principalmente serve a migliorare **l’aspetto estetico** e **presentare** la nostra pagina web per una migliore user experience (tramite animazioni, effetti, layout, posizionamento ecc…)

Un file CSS è costruito da tante regole, è composto principalmente da **2 PARTI**

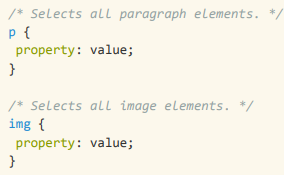
l’insieme di queste due: **proprietà + valore** crea una **🡪** **dichiarazione**

l’insieme dei SELETTORI e delle DICHIARAZIONI, si chiamano 🡪 **regole**

SELETTORI 🡪 ANDREMO A VEDERLI + AVANTI NELLO SPECIFICO

È quello che c’è **prima** delle { }, un selettore deve capire e dire al browser se selezionare un entità:

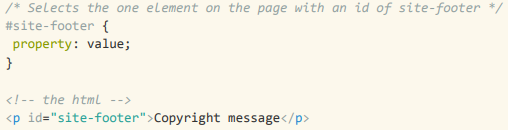
(**element selector**), e nel caso applicare le sue propietà, ne esistono di vari tipi, un esempio di default di un selettore che contiene d\* parametr\*, **l’element selector** è quello + basic:



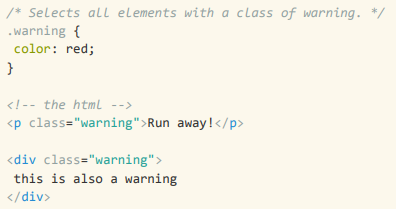
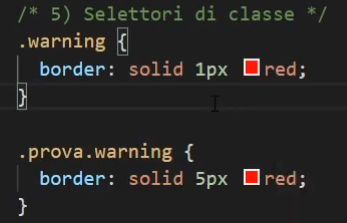
* **Selector relazionale** 🡪 andrà ad applicare le propreità a l’elemento **<strong>** in questo caso, ma solo se è **inglobato** all’interno dei tag 🡪 **<a> <li> <ul>** (se ci sono altri tag in mezzo, andrà comunque)



* **Selector ID** 🡪 applicarlo tramite **#** ad un unico elemento della pagina (dato che **è univoco**) il browser riesce a capire lo stesso anche se mettiamo **lo stesso ID** per + elementi, ma è sconsigliato.



* **Selector Class** 🡪 applicarlo tramite il . a uno o + elementi della pagina, un ottima soluzione per risolvere il problema degli ID che sono univoci, il selettore classe può essere applicato anche a + elementi con **tag diversi**, possiamo concatenare anche + classi insieme:



* **Selector groups** 🡪può comprendere e applicare i parametri ai tag normali <h1>, ma anche alle classi, id in maniera uniforme, suddividendoli con una ( **,** )
* **Selector Universale** 🡪 andrà ad abbinarsi con qualsiasi elemento tramite il ( **\*** )



PROPIETA

È tutto cio che è presente all’interno delle { }, le proprietà vengono **applicate dal browser** dopo che il selettore ha indicato gli elementi che deve modificare, ne esistono tantissimi, tra quali:

PROPIETA COLOR

settiamo il colore sul font, o anche sul background (bg), il browser può accettare i colori per varie strade con color name, **Hexadecimal value**, **RGB** value, **HSL** value

* **FONTI SUI COLORI**: [HTML Color Picker (w3schools.com)](https://www.w3schools.com/colors/colors_picker.asp) [147 Colors | CSS Color Names](https://147colors.com/)

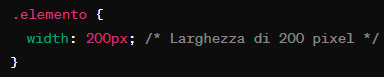
***216 colori*** sono i colori che si vedono nello stesso modo in TUTTI i browser, sono detti 🡪 **web safe color**

PROPIETA WIDTH

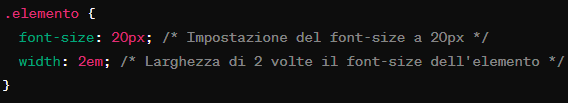
applicabile eslusivamente sugli elementi **block**, può avere vari unità di misura per misurare la lunghezza dell’elemento, analizziamo nel dettaglio le varie “unità di misura”:

- **px** 🡪pixel è la più piccola unità di colore e luminosità su uno schermo digitale. Quando si specifica una dimensione in pixel, si tratta di una misura assoluta e fissa.

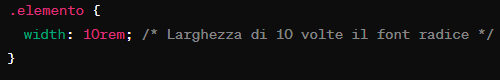
Se si imposta un elemento con width: 200px;, esso avrà una larghezza di 200 pixel **indipendentemente dalle dimensioni del dispositivo** o dello schermo dell'utente.



-**em** 🡪 se è 2 em, indica la grandezza x2 riferita **all’elemento stesso** (se lungo 16px, sarà lungo con em 32px)

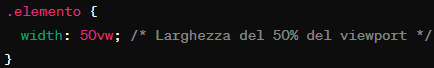


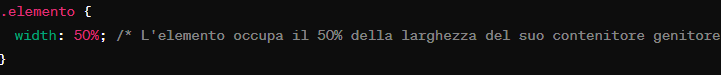
- **rem** 🡪 se è 3 rem, indica la grandezza x3 riferita **al font radice dell'elemento** (cioè il font di base dell'intera pagina HTML), (se è lungo 12 px, sarà lungo con rem 36px), nel caso non è stato stabilito dallo sviluppatore, si farà riferimento allo **“USER AGENT STYLESHEET”**



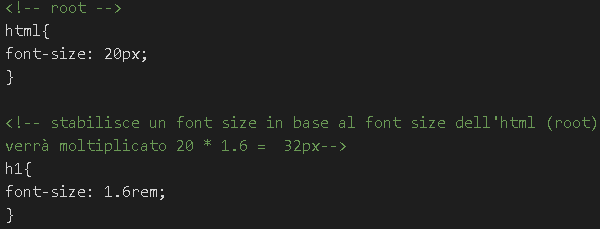
-**vw** / **vh** 🡪 sta x "**viewport width**" / "**viewport height**" (larghezza del viewport). Rappresenta una percentuale della larghezza totale della finestra del browser.

Se è 50vw, occuperà il 50% della **larghezza totale del viewport** (se il viewport ha una larghezza di 1000px, l’elemento avrà una lunghezza di 500px)



-**%** 🡪 utilizzata per definire la larghezza di un elemento in base a **una percentuale della larghezza del suo genitore**

Utilizzare **rem** per lavorare con la **root(html)** andremo a modificare gli elementi che hanno rem come UM



USER AGENT

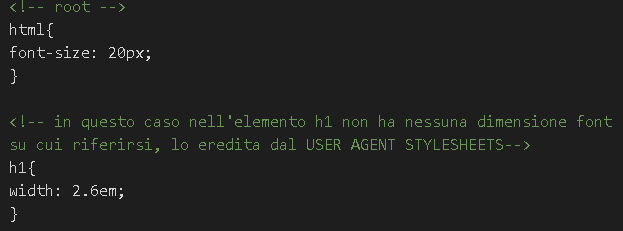
La combinazione di **sistema operativo**, **nome del browser**, **versione**, indica una stringa di identificazione inviata dal browser web di un utente a un sito web visitato.

sono utilizzate dai siti web per adattare i loro contenuti in base al tipo di browser e alle **capacità del dispositivo** dell'utente, per fornire una migliore esperienza utente:

USER AGENT STYLESHEET

è un **insieme predefinito di regole CSS** applicate automaticamente dal browser a tutte le pagine web, indipendentemente dallo stile specificato dallo sviluppatore web. Queste regole stabiliscono come gli elementi HTML di base dovrebbero essere visualizzati sullo schermo, garantendo una base uniforme per l'esperienza di navigazione.

Utile ad esempio in questo caso che non diamo un punto di riferimento al width dell’h1 (**em** sconsigliato)



PROPIETA FONT-FAMILY

La **font-family** proprietà CSS specifica un elenco prioritario di uno o più nomi di famiglie di caratteri e/o nomi di famiglie generiche per l'elemento selezionato.

Fonts customizzati sia per i browser + nuovi (slide 27) che per i browser + vecchi (slide 28)

***RENDONO PERO LE PAGINE + LENTE!!***

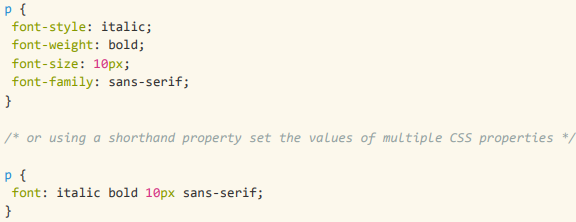
-**FONT-FAMILY** 🡪 indica la famiglia di font che vogliamo utilizzare, nel caso non è disponibile quel font, il browser deciderà lui quale usare in base all’**user agent stylesheets**.



-Il **serif** crea delle lineette sulle lettere (utilizzato per principalmente creare dei titoli)

-Il **sans-serif** non fa quello precedentemente detto (utilizzato per testo normale)

-**FONT SHORTAND** 🡪 per diminuire la fatica e le righe di codice (inutili):



**FONTS GOOGLE** 🡪 [Browse Fonts - Google Fonts](https://fonts.google.com/)|

**WEB SAFE FONTS** 🡪 [CSS Web Safe Fonts (w3schools.com)](https://www.w3schools.com/cssref/css_websafe_fonts.php)

ALTRE PROPIETA DEL CSS

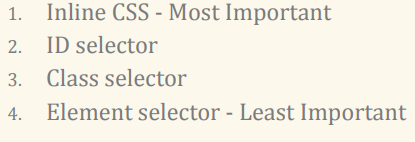
Le proprietà **+ comuni** 🡪 <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference>

TUTTE le proprietà 🡪 <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>

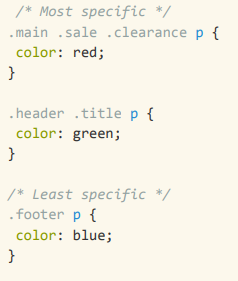
CONTROLLO SE IL BROWSER SUPPORTA QUELLE PROPRIETA 🡪 <https://caniuse.com/>

CSS CASCADE

Il browser assegna priorità diverse ai CSS a seconda del **tipo di selettore**, nel caso gli elementi abbiano lo stesso livello di importanza, l’ultimo sarà quello valido, ecco la graduatoria di importanza



Il browser assegna anche la priorità **in base alla specificità** della selezione I selettori più specifici hanno una **priorità** più alta --> [CLICCA QUI](file:///C:\Users\ICTS23-25.256\Desktop\HTML%20CSS\ESERCIZI\05-css-selectors\05-most-specific\05-most-specific.docx)



PSEUDO-CLASSES

Pseudo significa come se fosse qualcosa **cambia il comportamento in quel preciso momento**, ma realmente non lo è, lo applichiamo con **:** “due punti”:

**LISTA PSEUDOCLASSI** 🡪 [Pseudo-classes - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes)



CSS RESET

**Sovrascrive / rimuove** tutte le proprietà del **USER AGENT STYLESHEETS**, quindi applicando il codice di reset, il **<p>** avranno un aspetto uguale di un **<h1>**  (size =, weight =…).

\*remove all default browser styling to provide a clean slate,

\*eleminates most or all default browser styles, so requires to \*redefine styles for common HTML elements and require more \*maintenance to add back necessary styles.

È buona norma che lo sviluppatore aggiunga qualsiasi stile da zero.

* **CODICE RESET** 🡪 [CSS Tools: Reset CSS (meyerweb.com)](https://meyerweb.com/eric/tools/css/reset/)

CSS NORMALIZE

Mira a rendere lo stile del browser integrato **coerente** tra i browser, Questo significa che non tutti gli stili vengono azzerati, **ma vengono resi più consistenti e prevedibili**.

\* make browser render all elements more consistenly and inline

 \* whit modern standards whitout removing all styles, also

 \* provides a good base for most elements, requiring less

\*customization

 \* for standard elements

ex: Elementi come **H1-6** appariranno in grassetto, più grandi, ecc. in **modo coerente in tutti i browser**

* **CODICE NORMALIZE** 🡪 [Normalize.css: Make browsers render all elements more consistently. (necolas.github.io)](https://necolas.github.io/normalize.css/)

CSS DESTYLE

si occupa di ripristinare gli elementi al loro aspetto predefinito, senza alcun stile aggiuntivo, permettendo poi di applicare nuovi stili in modo più controllato e specifico.

\* Destyle is designed to remove only unwanted styles

 \* O problem, keeping other predefined browsers styles.

 \* In this way, Destyle offers greater flexibility than the

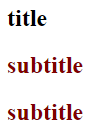
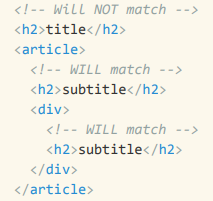
\* reset of the style

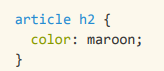
* **CODICE DESTYLE** 🡪 [Destyle.css - Opinionated reset stylesheet that provides a clean slate for styling your html. (nicolas-cusan.github.io)](https://nicolas-cusan.github.io/destyle.css/)

LEZIONE 6

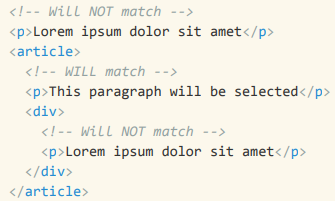
1. SELETTORI RELAZIONALI (combinatori) 🡪 [**vedere ESEMPI selettori**](file:///C:\Users\ICTS23-25.256\AppData\Roaming\Microsoft\ESERCIZI\05-css-selectors\01-css-diner\01-CSS-dinner.docx)

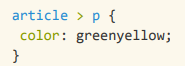
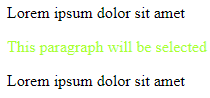
- **Descendant selector** (**SPACE**) 🡪 Seleziona un elemento che sono contenuti in un punto qualsiasi **all'interno di un elemento predecessore** identificato,in questo esempio abbiamo detto mi prendi TUTTI GLI **h2** se è contenuto all’intero di article



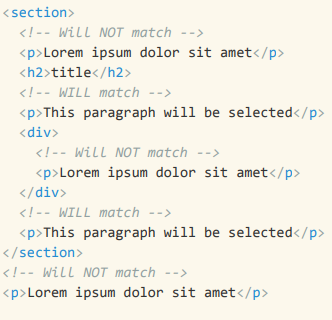
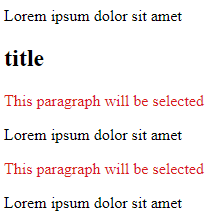


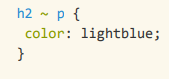
- **Direct child selector(>)** 🡪 Seleziona un elemento che **risiede immediatamente all'interno di un elemento padre identificato**, al contrario di prima non ci devono essere altri elementi di disturbo in mezzo tra padre e figlio, andremo a prendere ***esclusivamente*** solo i CHILD



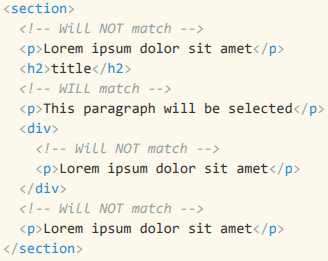


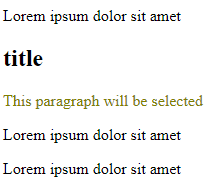
- **General Sibling selector (**~**)** 🡪 Seleziona un **elemento che segue un punto qualsiasi dopo l'elemento precedente** Entrambi gli elementi condividono lo stesso elemento padre

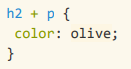




- **Adjacent Sibling Selecetor (**+**)** 🡪 Seleziona un **elemento che segue direttamente l'elemento precedente** Entrambi gli elementi condividono lo stesso elemento padre, quindi andiamo a prendere tutti i **p** che sono **FRATELLI**, **vicini** all’elemento h2





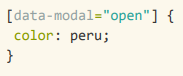


2. SELETTORI ATTRIBUTO 🡪 [**vedere ESEMPI selettori**](file:///C:\Users\ICTS23-25.256\AppData\Roaming\Microsoft\ESERCIZI\05-css-selectors\01-css-diner\01-CSS-dinner.docx)

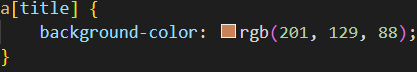
Alcuni Browser vecchi potrebbero non supportare questi selettori, ricodiamo che gli elementi HTML possono avere degli **attributi** come:

**class**  **id title** **target href data-modal alt**

**si distinguono in diverse categorie:**

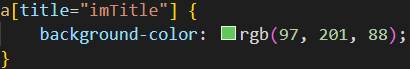


- **Attribute present selector 🡪** andiamo ad indicare di andare a prendere **quell’elemento che possiede quell’attributo**:





- **Attribute equals selector 🡪** in questo caso andiamo a specificare **UN DETERMINATO CONTENUTO ALL’INTERNO DI UN ATTRIBUTO**



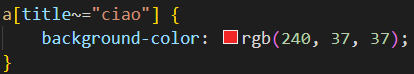


- **Attribute contains selector 🡪** andiamo a specificareun pezzo di un determinato contenuto in **qualsiasi posizione** si trovi (fine, mezzo, inizio) anche se è attaccata ad altre lettere e simboli, all’interno di un attributo TRAMITE IL **\***



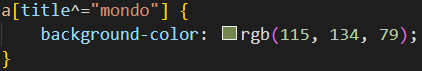


- **Attribute spaced selector 🡪**come nel precedente caso, andiamo a specificareun pezzo di un determinato contenuto in **qualsiasi posizione** si trovi (fine, mezzo, inizio), **MA DEVE ESSERE STACCATO DA ALTRE LETTERE E SIMBOLI** (space), all’interno di un attributo TRAMITE LA **~**





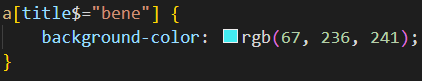
- **Attribute begins selector 🡪** andiamo a specificareun pezzo di un determinato contenuto che si trova **all’inizio** di esso (comincia), all’interno di un attributo TRAMITE IL **^**





- **Attribute ends selector 🡪** andiamo a specificareun pezzo di un determinato contenuto che è situato

**alla fine** di esso (termina), all’interno di un attributo TRAMITE IL **$**





- **Attribute hyphenated selector 🡪** andiamo a specificareun pezzo di un determinato contenuto che è contenuto e potrebbe essere SEGUITO DAL SIMBOLO DEL TRATTINO **-** **alla fine** di esso (termina), all’interno di un attributo TRAMITE L’ **|(or)**

3.0 PSEUDO-CLASSI : 🡪 [**vedere ESEMPI selettori**](file:///C:\Users\ICTS23-25.256\AppData\Roaming\Microsoft\ESERCIZI\05-css-selectors\01-css-diner\01-CSS-dinner.docx)

Vengono riconosciute dal prefisso dei **:**

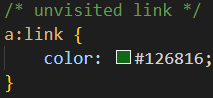
funzionano esattamente come gli attributi HTML ma NON sono scritte nel codice sorgente (HTML), **rappresentano lo stato di un elemento**, cambiano in base a ciò che fanno gli utenti o a come è strutturato il documento:

**LISTA COMPLETA DELLE PSEUDO-CLASSI** 🡪 [Pseudo-classes - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/Pseudo-classes)



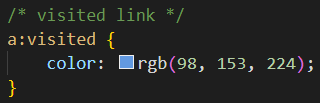
3.1 STATE PSEUDO-CLASSES 🡪 [**vedere ESEMPI selettori**](file:///C:\Users\ICTS23-25.256\AppData\Roaming\Microsoft\ESERCIZI\05-css-selectors\01-css-diner\01-CSS-dinner.docx)

- **unvisited link (:link)** 🡪 il link avrà quel comportamento fino a quando **non verrà visitato** almeno una volta:

****

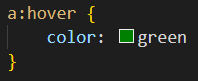
****

- **visited link (:visited)** 🡪 il link avrà quel comportamento quando **è stato visitato** almeno una volta:

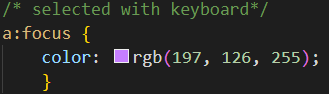
****

****

- **moused over (:hover)** 🡪 il link avrà quel comportamento quando andremo **sopra di esso con il cursore** del mouse 🡪 PER ESSERE UTILE :**HOVER** **VA MESSO DOPO** :**LINK** / :**VISITED**

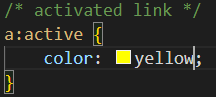




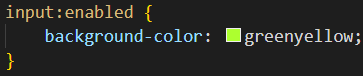
- **selected whit keyboard (:focus)** 🡪 comportamento applicato **quando l’elemento viene selezionato utilizzando la tastiera (TAB)**. Quando un link ha lo stato di "focus", significa che è stato selezionato, ma non necessariamente attivato:

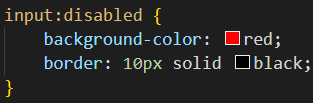


- **activated link (:active)** 🡪 comportamento applicato quando un link è attivato, ovvero quando **viene cliccato** con il puntatore del mouse 🡪 PER ESSERE UTILE :**ACTIVATE** **VA MESSO DOPO** :**HOVER**



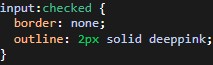
-**Enable elements (:enabled)** 🡪 Seleziona un input che si trova nello stato predefinito di **abilitato e disponibile** per l'uso:



-**disabled elements (:disabled)** 🡪 Seleziona un input a cui è associato **l'attributo disabled** all’interno del file HTML:



-**checked elements (:checked)** 🡪 applicato agli input di type **RADIO** oppure **CHECKBOX**, avranno quel determinato comportamento quando vengono selezionati dall’utente

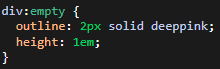


3.2 STRUCTURAL & POSITION PSEUDO-CLASSES 🡪 [**vedere ESEMPI selettori**](file:///C:\Users\ICTS23-25.256\AppData\Roaming\Microsoft\ESERCIZI\05-css-selectors\01-css-diner\01-CSS-dinner.docx)

Queste pseudo-classi **consentono di selezionare elementi in base alla loro posizione all'interno della struttura** del documento HTML:

**X ESERCITARSI E VEDERE ESEMPI REALI** 🡪 [Useful :nth-child Recipes | CSS-Tricks - CSS-Tricks](https://css-tricks.com/useful-nth-child-recipies/)

-**:empty** 🡪 rappresenta qualsiasi elemento **che non ha figli**:





-**:first-child** 🡪 <https://developer.mozilla.org/en-US/docs/Web/CSS/:first-child>

  rappresenta **il primo elemento** di un gruppo di elementi fratelli **🡪** (ul **li1**  li2 … )

-**:last-child** 🡪 <https://developer.mozilla.org/en-US/docs/Web/CSS/:last-child>

rappresenta **l'ultimo elemento** di un gruppo di elementi fratelli 🡪 (ul li1 li2 **li3**)

-**:only-child 🡪** <https://developer.mozilla.org/en-US/docs/Web/CSS/:only-child>

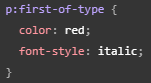
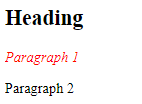
Seleziona l'elemento che **è l'unico figlio all'interno del suo genitore**, cioè non ha altri fratelli.

(ul **li1**) (ul **li1 li2 li3**)

-**:first-of-type** 🡪 [:first-of-type - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/:first-of-type)

rappresenta il **primo elemento del suo tipo** all'interno del suo genitore, tra un gruppo di

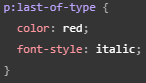
elementi di pari livello.



-**:last-of-type** 🡪 [:last-of-type - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/:last-of-type)

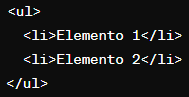
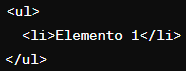
rappresenta **l'ultimo elemento del suo tipo** all'interno del suo genitore, tra un gruppo di

elementi di pari livello.



-**:only-of-type** 🡪[:only-of-type - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/:only-of-type)

rappresenta gli elementi che **sono l'unico elemento di quel tipo** all'interno del loro genitore:



-**:nth-child()** 🡪 seleziona **gli elementi figli in base alla loro posizione** tra TUTTI gli elementi fratelli all'interno di un elemento genitore

<https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-child>

-**:nth-last-child()** 🡪 seleziona **gli elementi figli in base alla loro posizione** tra TUTTI gli elementi fratelli, all'interno di un elemento genitore, !! MA AL CONTRARIO QUI **PARTIAMO DAL FONDO !!**

<https://developer.mozilla.org/en-US/docs/Web/CSS/:nth-last-child>

4.0 PSEUDO - ELEMENTI : :

Sono **elementi dinamici** che NON esistono nell'albero del documento.

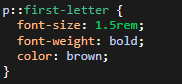
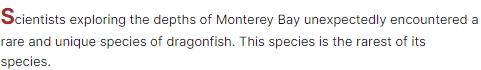
Se utilizzati all'interno dei selettori, consentono di stilizzare parti uniche della pagina, è possibile **utilizzare un solo pseudo-elemento all'interno di un selettore** in un determinato momento

Viene riconosciuto dal prefisso dei due punti doppi ( **: :** )

4.1 PSEUDO-ELEMENTI TESTUALI

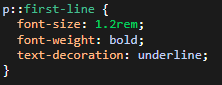
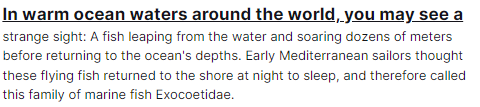
**-::first-letter**🡪 Selezionare la **prima lettera di testo** all'interno di un elemento di tipo “**block**”

[::first-letter - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/::first-letter)



**-::first-line** 🡪 Seleziona Identifica **la prima riga di testo** all'interno di un elemento di tipo “**block**”

[::first-line - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/::first-line?retiredLocale=it)

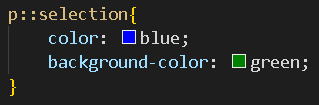
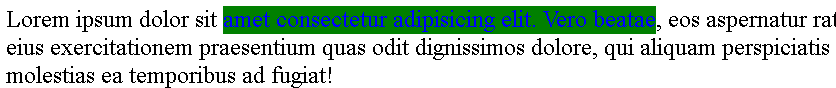


4.2 PSEUDO-ELEMENTI CONTENUTO

4.3 PSEUDO-ELEMENTI FRAMMENTO

permettono di stilizzare la parte di un documento che un **utente ha selezionato o evidenziato** utilizzando il proprio browser:

**-::selection** 🡪mira alla **porzione** di un documento che è stata **selezionata** dall'utente, potresti usarlo per far risaltare il testo selezionato cambiandone il colore dello sfondo e il colore del testo:



**-::moz-selection** 🡪 ha la stessa funzione di **::selection**, ma è specificamente utilizzato per Firefox per garantire la compatibilità e la stilizzazione uniforme su diversi browser.

SPECIFITY

A una dichiarazione CSS viene **applicato un peso**, determinato dal numero di ciascun tipo di selettore

**PER VEDERE IL SUO FUNZIONAMENTO** 🡪 [CLICCA QUI](../ESERCIZI/05-css-selectors/05-most-specific/05-most-specific.docx)

LEZIONE 7

Installazione di un live server tramite **cmd node js** 🡪 link di stefano cherio

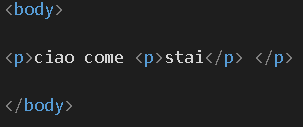
ELEMENTI BLOCK | INLINE

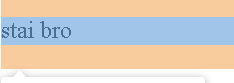
Gli elementi HTML possono essere divisi in **due categorie principali** in base al loro comportamento di layout: **elementi block** e **elementi** **inline**:

-**ELEMENTI BLOCK** 🡪 Gli elementi block **occupano tutto lo spazio orizzontale** disponibile e vanno a capo, rendendo il successivo elemento disposto su una nuova riga.

Gli elementi block per quanto riguarda il box-model: **width** **height (content) padding border margin**





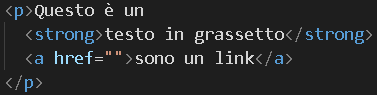


-**ELEMENTI INLINE** 🡪 Gli elementi inline occupano **solo lo spazio necessario orizzontalmente** e non vanno a capo automaticamente dopo di loro, l’elemento successivo viene **disposto accanto** all'interno dello stesso flusso di testo, senza andare a capo.

Gli elementi inline per quanto riguarda il box-model: **width**  **height (content) padding border margin**

**margin 🡪 sono applicabili solo sui lati SINISTRA DESTRA**





SLIDE GRUPPO : [here](06b-hc-css-grouping-and-box-model.pdf) [gioco](https://box-model-playground.vercel.app/) SLIDE PROF : [here](06b-hc-css-grouping-and-box-model.pdf)

-PROPRIETA DISPLAY

è un modo potente per controllare come gli elementi HTML vengono visualizzati nel browser, andiamo a vedere come si può applicare:



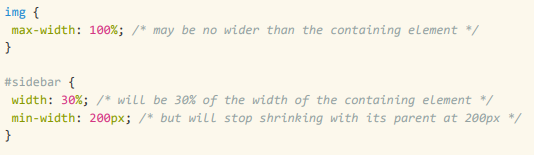
display: block; 🡪 l’elemento inline diventa block

display: inline-block; 🡪 un elemento inline prende le caratteristiche di un elemento block

-OTHER CSS PROPERTIES

**Width** **Height** 🡪 non funzionano e non sono applicabili **sugli elementi INLINE**

**Min max 🡪** stabilisce una dimensione massima e minima per quanto riguarda **Width** **Height**

****

**Calc( … ) 🡪** È possibile combinare diverse unità come percentuali, pixel, ems e altro ancora

-FLOW

-BOX MODEL:

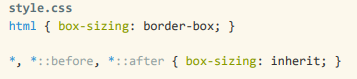
PADDING

BORDER

MARGIN

BOX-SIZING

Da applicare sempre all’interno del nostro codice:



-MARGIN-COLLAPSE

-SEMANTIC ELEMENTS

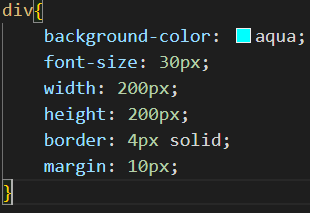
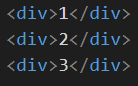
[HTML Semantic Elements (w3schools.com)](https://www.w3schools.com/html/html5_semantic_elements.asp)

LEZIONE 8 🡪 [SLIDE PROF](07-hc-positioning-and-display.pdf)

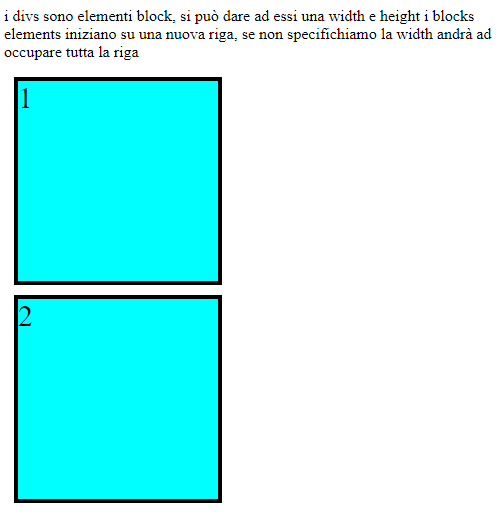
DISPLAY 🡪 [CSS Layout - The display Property (w3schools.com)](https://www.w3schools.com/css/css_display_visibility.asp)

viene utilizzata per specificare il **comportamento del rendering di un elemento HTML**, come questo elemento viene visualizzato nel layout della pagina.

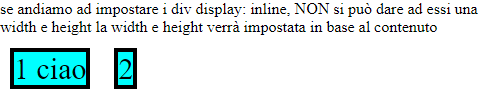
Esistono principalmente 4 opzioni per questa propietà:



-**display: BLOCK** 🡪 occupano **tutta la larghezza disponibile del loro contenitore padre**, e vengono visualizzati come **blocchi**, con un'interruzione di riga prima e dopo di essi:

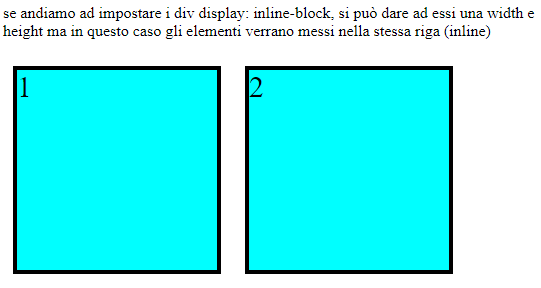


-**display: INLINE** 🡪 occupano **solo la larghezza necessaria per il loro contenuto**, e non causano interruzioni di riga prima o dopo di loro:

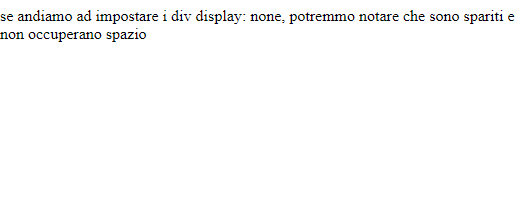


-**display: INLINE-BLOCK** 🡪 combina **caratteristiche degli elementi BLOCK e INLINE**, l’elemento non causa un'interruzione di riga, ma può avere una larghezza e un'altezza impostate come un elemento di blocco.

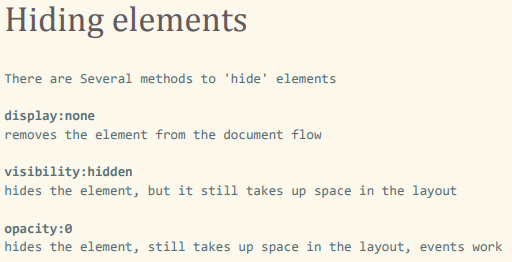
**è utile quando si desidera combinare la disposizione in linea con la capacità di impostare dimensioni specifiche:**

****

-**display: NONE** 🡪 **Nasconde l'elemento dal layout**, come se non esistesse nel **DOM**, di conseguenza, NON PRENDERA SPAZIO NELLA PAGINA E NON VIENE VISUALIZZATO!:



Altre note importanti simili ma con effetti diversi dal display: none

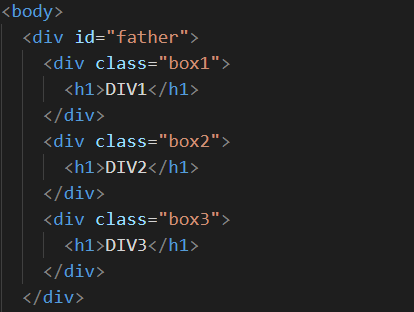


BACKGROUND

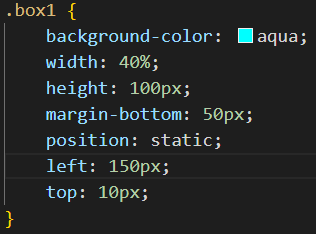
File CSS con spiegazioni ed esempi riguardo le **background propiety** 🡪 [**background.css**](../ESERCIZI/07-positioning-and-display/00-presentation-background/CSS/style.css) **gioca!**

POSITIONING 🡪 [esercitati qui!](../ESERCIZI/07-positioning-and-display/00-giocare-position/CSS/style.css)

Il posizionamento (positioning) in CSS è una proprietà che consente di **controllare il posizionamento degli elementi HTML nel documento**. PER DEFAULT TUTTI GLI ELEMENTI HANNO POSIZIONE STATIC

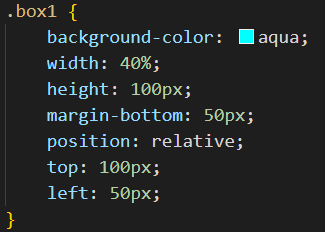
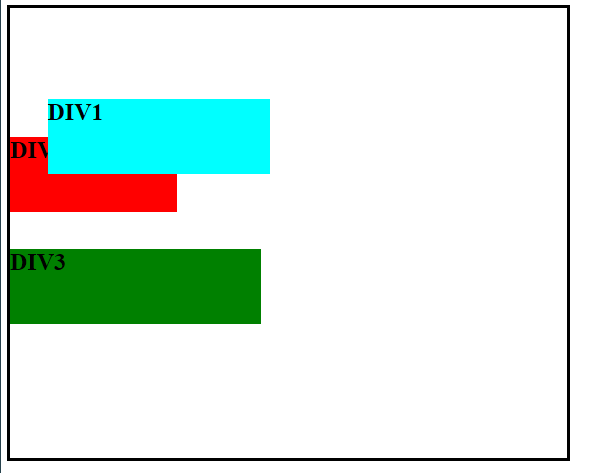


-**position: STATIC** 🡪 l’elemento **non potrà MAI cambiare la sua posizione**, rimarrà nella sua posizione di nascita di dove è stato creato nel flusso del documento, come vediamo se proviamo a modificarla:



-**position: RELATIVE** 🡪 l'elemento **viene posizionato secondo il normale flusso del documento**, esso NON INFLUISCE SUL FLUSSO DEL DOCUMENT e sugli elementi che lo circondano, e può essere **spostato** come vogliamo, **PARTENDO dalla sua posizione originale** usando le proprietà:

**top**, **right**, **bottom**, **left** (possono avere sia valori POSITIVI che NEGATIVI):



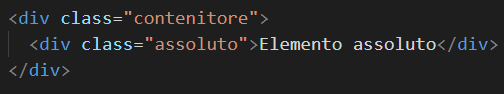
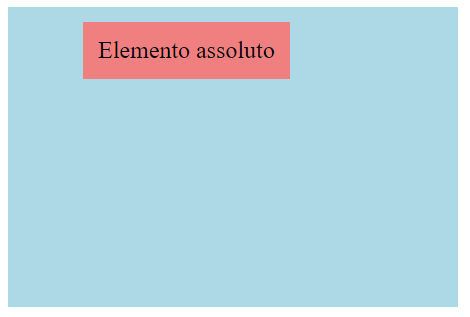
-**position: ABSOLUTE** 🡪 consente di posizionare un elemento **rispetto al suo GENITORE posizionato** **più vicino**. Esso INFLUISCE SUL NORMALE FLUSSO DEL DOCUMENTO.

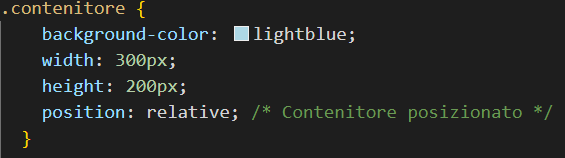
Se NON ci sono antenati posizionati (ossia, antenati con **position diversa da static**), l'elemento viene posizionato rispetto al contenitore iniziale OSSIA IL **TAG HTML**(viewport).

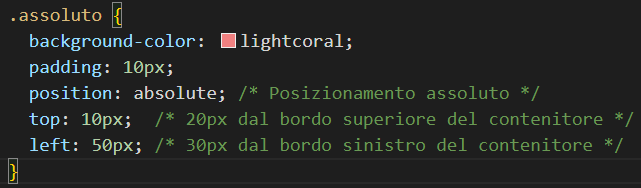
ANDIAMO A VEDERE UN ESEMPIO DOVE 🡪 **POSIZIONAMENTO RISPETTO AL CONTENITORE + VICINO:**

in questo esempio, l'elemento con **classe .assoluto** è posizionato 20px sotto il bordo superiore e 30px a destra del bordo sinistro del suo contenitore posizionato, che ha la classe **.contenitore**.

Poiché .contenitore ha position: relative, diventa il riferimento per il posizionamento dell'elemento assoluto.



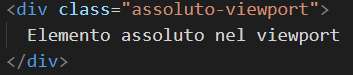
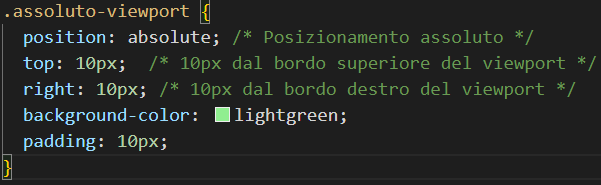
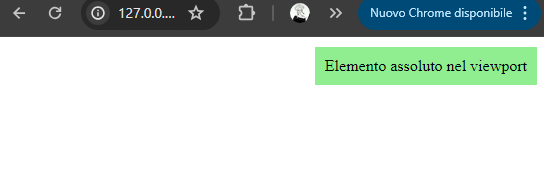




ANDIAMO A VEDERE UN ESEMPIO DOVE 🡪 **POSIZIONAMENTO RISPETTO AL VIEWPORT:**

In questo esempio, l'elemento con **classe .assoluto-viewport** è posizionato 10px sotto il bordo superiore e 10px a sinistra del bordo destro del viewport, perché NON CI SONO ANTENATI posizionati!

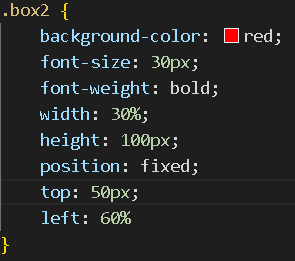
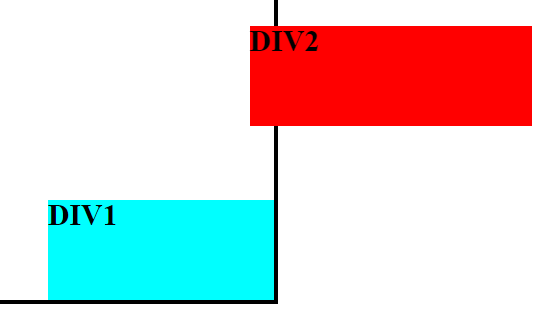
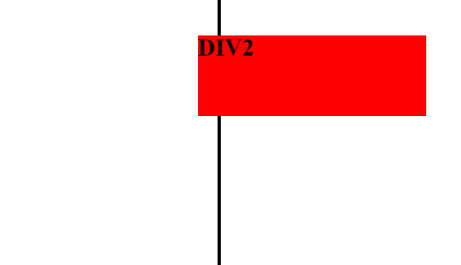
Quindi, il viewport diventa il riferimento per il posizionamento.



[GIOCA QUI CON POSITION ABSOLUTE](../ESERCIZI/07-positioning-and-display/00-giocare-position-ABSOLUTE/CSS/style.css)

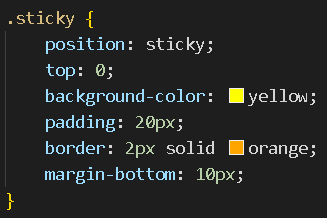
-**position: FIXED** 🡪 viene utilizzata per posizionare un elemento ***rispetto alla finestra del browser***, e NON rispetto al contenitore padre o alla pagina stessa.

Ciò significa che l'elemento rimarrà **fisso nella posizione specificata**, anche quando si scorre la pagina:

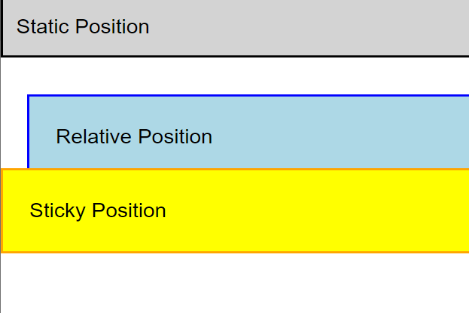


assicurare di **testare il comportamento** degli elementi fissi su diversi dispositivi e dimensioni dello schermo per garantire una buona esperienza utente.

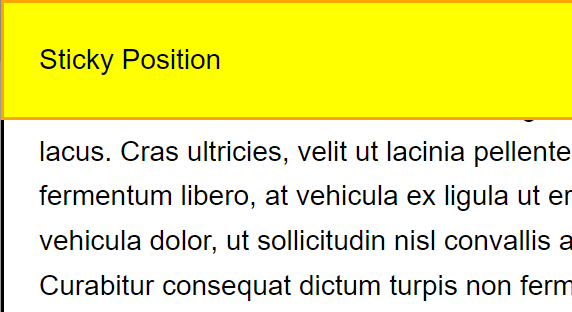
-**position: STICKY** 🡪 **combina aspetti delle posizioni relative e fixed**.



Un elemento con position: sticky 🡪 **è trattato come relative** finché non scorre oltre un punto specificato cioè segue il normale flusso del documento.



dopodiché viene trattato come **fixed** "appiccicoso". Questo è utile per elementi che devono rimanere visibili all'interno del loro contenitore durante lo scorrimento:



Z-INDEX

[Using z-index - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_positioned_layout/Understanding_z-index/Using_z-index)

LEZIONE 9 🡪 [SLIDE PROF](08-hc-float.pdf)

INTRODUZIONE AL FLOAT + GIOCO LINK

<https://developer.mozilla.org/en-US/docs/Web/CSS/float>

<https://developer.mozilla.org/en-US/play>

Sono stati utilizzati anche per **creare layout** con più colonne, perché all'epoca non c'erano buoni modi per creare un layout complesso.

***posiziona un elemento sul lato sinistro o destro del suo contenitore, consentendo al testo e agli elementi in linea di avvolgerlo.***

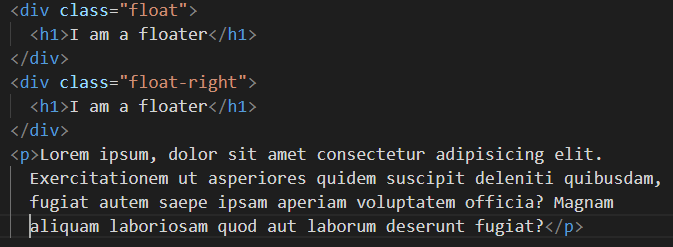
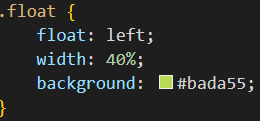
FLOATING ELEMENTS

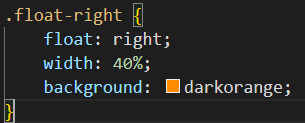
**Quando applichiamo il float**, ad esempio a un <div>, questo NON si comporterà più come un elemento **block**, **ma diventerà un elemento float** (simile a inline-block).

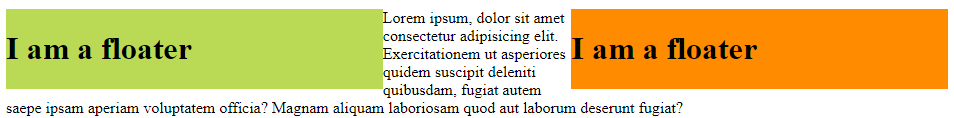
Ad esempio:

se **creiamo due <div>** con float **OPPOSTI** (uno a sinistra e uno a destra), e poi **aggiungiamo un <p>,** il testo del <p> **potrebbe incastrarsi in mezzo ai due <div>,** a meno che non venga chiaramente specificata una larghezza (width) sufficiente.

Questo accade **perché il float permette agli elementi di fluttuare a sinistra o a destra** rispetto agli altri elementi, modificando il normale flusso del documento.







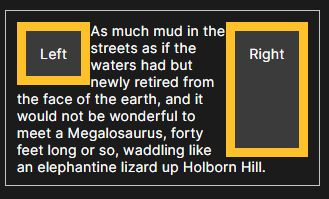
CLEAR

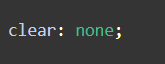
https://developer.mozilla.org/en-US/docs/Web/CSS/clear

Imposta se un elemento **deve essere spostato al di sotto degli elementi mobili** (deselezionati) che lo precedono. La proprietà **clear** si applica agli elementi mobili e non mobili:

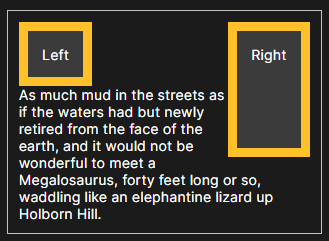
-----------------------------------------------------------------------------------------------------------------------------------------------

indica che l'elemento **NON** viene spostato verso il basso per cancellare gli elementi mobili passati.





indica che l'elemento viene spostato verso il basso per cancellare i float successivi a sinistra.



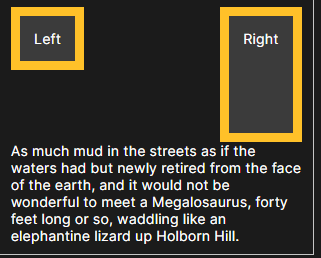


Indica che l’elemento viene spostato verso il basso per eliminare i float a destra





 indica che l'elemento viene spostato verso il basso per superare sia i float di sinistra che quelli di destra.





LEZIONE 10 🡪 [SLIDE PROF](09-hc-responsive-web-pages.pdf)

1.CONTENT FIRST

Teoria, guarda le [slide](09-hc-responsive-web-pages.pdf)

2.MOBILE FIRST

Teoria, guarda le [slide](09-hc-responsive-web-pages.pdf)

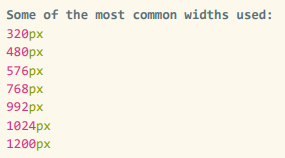
3.RESPONSIVE DESIGN

Teoria, guarda le [slide](09-hc-responsive-web-pages.pdf)

4.BREAKPOINTS

I breakpoints si chiamano “**punti di interruzione**”, sono **regole CSS** che permettono di adattare il layout di una pagina web a diverse dimensioni dello schermo.

Utilizzando i breakpoints, è possibile applicare stili specifici **per diverse larghezze di schermo**, rendendo il design della pagina responsive e ottimizzato per vari dispositivi come desktop, tablet e smartphone:



IL **MINIMO** BREAKPOINT PER RENDERE LA PAGINA RESPONSIVA è **320px**

Non esiste uno standard definito per le larghezze da scegliere come target nelle media query

● Qualsiasi insieme ragionevole di incrementi è sufficiente per indirizzare la maggior parte dei dispositivi

● L'obiettivo è quello di avere punti di interruzione sufficienti per scegliere come target smartphone, tablet, laptop e desktop

5.VIEWPORT META TAG

Cosa è la [viewport](https://www.w3schools.com/css/css_rwd_viewport.asp)?

Definisce il modo in cui un sito **deve essere visualizzato in un browser Web per dispositivi mobili**

● Fa in modo che le media query funzionino come previsto

● Deve sempre includere la seguente riga di HTML nell'intestazione di **tutti i documenti**



-**device-width** 🡪 imposta la larghezza del viewport alla larghezza **dello schermo del dispositivo**, assicurando che il contenuto non sia ridimensionato in modo non corretto.

-**initial-scale** 🡪 assicura che lo zoom **NON** venga mai applicato, e che il layout venga sempre visualizzato in scala **1:1**

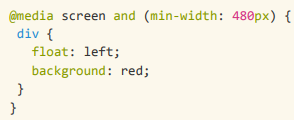
6.MEDIA QUERIS

Tramite le media queris, consentiamo la personalizzazione di pagine web per dispositivi specifici ○ telefoni cellulari, tablet, desktop, ecc..

Il codice HTML **NON** viene modificato, **solo lo stile CSS**

Una media query è un'espressione logica: **true** o **false**, Se una media query è vera, le regole correlate vengono applicate al dispositivo di destinazione

ecco un esempio banale di media query:



7.RESPONSIVE IMAGES

LEZIONE 11 [SLIDE PROF](10-hc-scss.pdf)

Tutto sulle slide del prof, esempi sulla folder 00-try

LEZIONE 12 [SLIDE PROF](11-hc-shadows-gradients-animations.pdf)

BOX-SHADOW

Esempi pratici qui 🡪 [clicca](../ESERCIZI/11/CSS/style.css)

[box-shadow - CSS: Cascading Style Sheets | MDN (mozilla.org)](https://developer.mozilla.org/en-US/docs/Web/CSS/box-shadow)

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Immagine che contiene viola, Lilac, violetto, Magenta

Descrizione generata automaticamentePossiamo anche volendo **mettere + strati** uno sotto l’altro di ombre, separando i valori di ogni shadow con la 🡪 “**,**”, come in questo esempio:

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente