<https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web>

**HTML - HyperText Markup Language**

file salvato in root cartella; + cartella images per immagini, styles per CSS e scripts per JavaScript.

index.html

**<elemento attributo=””>**

<p>My name is W</p> - esempio

<!-- comment here -->

Intestazione:

<!DOCTYPE html>

<meta charset="utf-8" />

<meta name=”author” content=”” />: autore della pagina;

<meta name=”description” content=”” />: tema della pagina, per SEO;

<link rel="icon" href="favicon.ico" type="image/x-icon" />: aggiunge iconcina sulla tab;

ul - unordered list; ol - ordered list; li - elementi nella lista

<h1-h2-h3-h4-h5-h6/> <p>

<a href=””></> = anchor con hypertext reference, inline - anche immagini, video… - con title=”” compare scritta se si hovera; per indicare il link nella cartella del sito, usare /, per tornare su: ../;

con id=”” e href=”#” si può ancorare un elemento della pagina e cliccandoci, ricondurlo;

URL assoluti: intero percorso file; relativi: solo la parte diversa dalla posizione di index.html;

essere specifici nella formulazione del link, non ripetendosi e essendo concisi sul contenuto;

attributo download in <a> per nominare il file da scaricare;

<a href=”emailto:aaa@bb.com”>

<strong>: grassetto-bold, inline;

<em>: italics-emphasis, inline;

meglio non usare <b>;<i>;<u>;: solo in casi specifici come errori di spelling (u); parole scientifiche o straniere (i) e tasti da tastiera

block elements: vanno a capo;

block-inline HTML diversi da CSS: in CSS sono proprietà di visualizzazione, mentre in HTML sono immutabili caratteristiche intrinseche.

in tag che non inglobano testo da mostrare, si può chiudere con />;

attributi: class=”aaa” spazio id=”bbb” …;

attributo title: quando hover, mostra di cosa si tratta il link; target:”\_blank” manda il link in una nuova scheda;

required, disabled: boolean attribute, che non necessitano di specificazioni;

“” = ‘’, quotes si equivalgono; meglio metterle per evitare ambiguità nella lettura della pagina;

Special Characters:

<= &lt; >= &gt; “= &quot; ‘= &apos; &= &amp; spazio= %20;

in URL: ?= separa principale da settaggi; &= separa i vari settaggi;

descrizioni:

<dl>: description list; per iniziare la descrizione;

<dt>: description term: inquadra il termine;

<dd>: description definition: definizione del termine; anche multiple dd per un dt;

<blockquote cite=”url”>: per effettuare una citazione sul pezzo selezionato; diventa un blocco indentato;

<q cite=”url”>: citazioni inline; diventa <<...>>;

<cite>: rende testo in corsivo, si può aggiungere un <a href> con il link (poco utile);

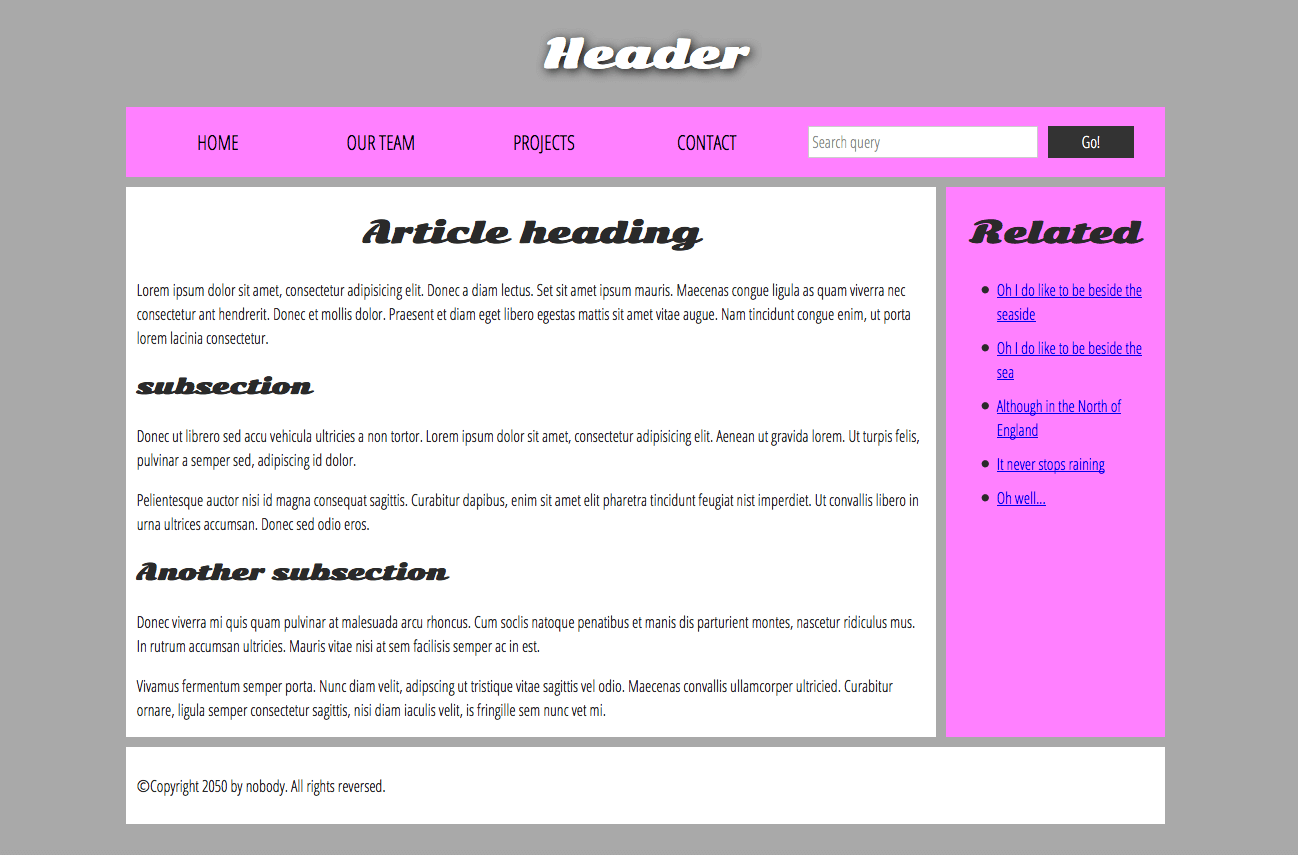
<abbr title=””>: evidenziare le abbreviazioni; prima volta è bene spiegarle con il resto del paragrafo;

<address> per evidenziare un indirizzo, bene anche in aggiunta a ul e p;

<sup> + <sub>: superscript e subscript, servono per creare i numeri ordinali (st, nd, rd, th) e formule chimiche;

Altri elementi per evidenziare in HTML: <code>, <var>, <pre> (mantiene lo spazio del paragrafo come in codice), <kbd> (tasti da tastiera), <samp> (output di programma);

<time datetime=””>per evidenziare date e ore, meglio nel formato yyyy-mm-dd hh:mm;

Sezioni di un documento HTML:

<header>: fascia con logo, nome del sito che rimane sempre in alto;

<nav>: barra di navigazione, consistente tra le pagine; serve per collegare le varie sezioni di un sito; certe volte unito assieme all’header;

<main> content: contenuto principale, con dentro vari <article>, <section>, e <div>;

<aside>: sidebar con contenuti secondari, come bio autore, oppure un’altra barra di navigazione;

<footer>: fascia in basso con informazioni su copyright, contatti ecc… simile a header;

<article>: racchiude un blocco di informazioni con senso proprio; <section>: simile ad article, ma usato più per blocchi con tema comune o simili funzionalità;

<span>: elemento inline per dare caratteristiche particolari tramite CSS, non semantico;

<div>: elemento block per dare caratteristiche particolari tramite CSS, non semantico;

<br />: forza andare a capo, utile per liste o poesie;

<hr />: separa i due elementi tramite una lunga linea orizzontale, per separare i temi;

Debug: HTML più facile di altri linguaggi; due tipi di errori: sintassi (mancano “”, <> …); oppure logici (scrittura corretta ma errore di altro tipo, più concettuale); HTML non soffre di errori di sintassi, dal momento che è permissivo: mostra sempre qualcosa, anche se per l’utente è sbagliato! HTML corregge e interpreta alcuni errori; con Analizza si può modificare la pagina web; si può anche modificare il CSS;

<https://validator.w3.org/> - per validare e trovare errori nel HTML;

Creare pagina HTML:

avere bene in mente le parti comuni, come header, nav e footer; pensare e schematizzare i contenuti da scriverci per ogni pagina; lavoro di preparazione è essenziale; fare schema di una mappa del sito;

Multimedialità

<img src="" alt="description" >: elemento vuoto, non prende il </> e non si può inserire elementi dentro; src è la fonte dove si trova l’immagine; alt è un testo alternativo se l’immagine non viene caricata, usato anche in ricerca per immagini; meglio dare nomi descrittivi a immagini, per SEO; non linkare immagini da altri siti, attenzione a copyright; img è un elemento il cui contenuto non subisce modifiche da parte di CSS, si chiama replaced element; carino elencare height e width in img, anche se per modificarla è meglio CSS; title crea una minididascalia da mostrare quando si hovera sopra immagine;

<figure> contenitore per elementi multimediali e <figcaption>, la descrizione dell’elemento in HTML normale: da usare per collegare in un unico blocco semantico la foto e la descrizione;

con CSS background-image si può mettere immagine come sfondo, solo come decorazione;

<video>: per inserire video, anche da Youtube o altro; richiede attributo src=”fonte”; controls: interfaccia per gestire il video, metterlo in pausa e controllarne il volume, con browser o APIs. tra le <></> si definisce il fallback, in caso il browser non supporti la riproduzione di video, conveniente inserire lì il link: fallback scritto tra <p> e incluso in <video> come testo; per supportare vari formati e dispositivi è possibile separare il parametro src in un tag autonomo, all’interno di video, chiamato <source src=”” type=””> dove elencare i vari formati video; attributo type indica il tipo mime di file video: il browser skipperà il formato che non riconoscerà, evitando di sprecare risorse inutili; altri attributi video: height-width (ma il video mantiene aspect ratio, al contrario di img); autoplay-loop-muted (meglio non usarli); poster=”url”:thumbnail, immagine prima di partire play; preload per effettuare un caricamento durante la riproduzione (none, auto o metadata);

<audio>: funziona come video, ma non supporta width-height e poster;

<tracks>: per inserire la trascrizione del testo, all’interno dei tag <video>, <audio>, dopo <source>; attributi: kind=”subtitles (traduzioni), captions (trascrizioni) o descriptions”, src=””, srclang=”lingua del transcript per il browser”, label=”lingua per gli utenti”;

esempio: <video controls … height width><source src=”” type=””><tracks …>Fallback</video>

<iframe>: pagine web, specialmente utile per linkare video su youtube o mappe google, ads struttura come <video> attributo sandbox: richiede misure di sicurezza maggiori;<iframe> pone problemi di sicurezza, meglio sempre impostare sandbox (che restringe e quarantena il campo); <embed>, <object> sono altri tag per includere file, come pdf. Oggi non più molto usati, evitare.

grafica vettoriale: mantiene aspect ratio, e non sgrana: consigliata. Raster images: jpg, png, gif, bmp … ogni pixel è fisso; vector images: scalano la risoluzione tramite algoritmo; svg è un formato xml, come un HTML; le parole scritte in svg sono lette in SEO; da aggiungere come <img height=”” width=””>; si può anche aggiungere come testo semplice, basta copiare il codice e includerlo nel tag <svg></svg>, lo si puô modificare anche con CSS; anche possibile includerlo in <iframe>; ma non consigliato;

Immagini responsive a schermi differenti: problema di direzione artistica (troppe immagini diverse per accomodare schermi diversi); cambio risoluzione (immagine piccola per risparmiare banda si ingrandisce e si sgrana per accomodare schermi grandi); svg e immagini vettori sono utili se semplici, meglio jpg se complesse; di solito CSS meglio per problemi di responsività;

problema di cambio risoluzione: aggiungere attributi srcset e sizes al tag <img>; srcset con tutte le diverse foto grandi e piccole; sizes con condizione - (max-width: xy) dimensioni foto da visualizzare; src e alt rimangono di default;

<img srcset=”abc.jpg 400w, bce.jpg 800w” sizes=”(max-width: xypx) 400px, 800px” src=”” alt=””>

srcset composto da nome foto, uno spazio e dimensione intrinseca in w; sizes composto da condizione media (max-width: xy), spazio e lunghezza slot della foto da occupare, se condizione è falsa, html va al successivo, fino a quello di default; oltre a px, si può usare vw: viewport width, un valore in relazione al viewport; stessa operazione possibile anche senza sizes ma usando moltiplicatore x;

problema di direzione artistica: usare immagini diverse per accomodare schermi diversi, con <picture>, che funziona come <video>, inglobando diversi <source>

<picture><source media=”(max-width: xy)” srcset=””><img src=”” alt=””></picture>

contiene come attributo media, che definisce la condizione, srcset con il file da disporre nel caso la condizione sia vera; <img> è in questo caso il fallback, nel caso nessuna condizione si avvera; se attributo media, non usare size; CSS e JavaScript sono caricati dopo HTML, rendendo necessario usare queste tecniche prima di altri formati; type attributo aggiungibile, per far rigettare al browser formati non supportati;

HTML Tabelle:

<table>contenuto tabella</table>; <td> : table data, ovvero celle; <tr> : table row, per racchiudere le celle e iniziare nuove righe; <th> : table header, ovvero intestazione riga-colonna (nella riga delle intestazioni di colonna, sono tutti <th>); attraverso l’attributo scope=”col”, si può legare la riga/colonna <th> al contenuto <td>, utile per l’accessibilità, stessa cosa con <th scope=”row”> per prima riga, oppure “colgroup”/”rowgroup” per celle che prendono più spazio; attributo colspan e rowspan permettono alla cella di prendere varie lunghezze; per applicare uno stile alle colonne esiste il tag <col>; sintassi: subito dopo <table> creare gruppo <colgroup> e poi ogni <col> fino a quello che si vuole modificarne lo stile; in caso contrario lo stile si applica al primo col; <caption> sotto table permette di fornire una didascalia della tabella;<thead>, <tbody>, <tfoot> segnano un header, body e footer di una tabella, di per sé niente di speciale, bensì permettono di avere nuovi elementi per CSS; <thead> dopo <colgroups> se presenti; si può anche inserire tabelle in tabelle;

XML: simile a HTML e SVG ma permette di creare e definire i propri tag;

<?xml version="1.0" encoding="UTF-8"?> - intestazione;

<?xml-stylesheet type="text/xsl" href="transform.xsl"?> - con XSL lo si può trasformare in html;

**CSS - Cascade Style Sheet**

per modificare lo stile e il layout di html; di default CSS ne applica uno di base, che si può modificare con CSS; CSS è case insensitive;

styles.css

in HTML per collegarlo con tre metodi:

<link href="./styles/styles.css" rel="stylesheet" type=”text/css” /> - Metodo 1;

<head><style> ogni proprietà css </style></head> - Metodo 2 (direttamente in html, poco efficiente per molte pagine);

<elemento style=””>abc</elemento> - Metodo 3 (inline, per modifiche specifiche)

/\* comment here \*/

CSS procede a cascata: regole più specifiche e/o successive sovrascrivono le precedenti in caso di conflitto! specifico>successivo però;

p, h1 {color: red;} - esempio; p, h1 sono **selettori** che aprono la regola; all’interno di {} ci sono dichiarazioni: color è la **proprietà**, mentre red è il suo **valore**; l’insieme delle dichiarazioni si chiama declaration blocks; mentre tutto il blocco CSS con il selettore: CSS ruleset; nomi delle proprietà non hanno mai spazi. Spazi come in HTML per maggiore leggibilità ma ininfluenti nel software;

per cercare online la i valori delle proprietà: mdn proprietà!!!

valore none: elimina lo stile di default - meglio guardare le pagine di referenze per tutti i valori;

selettori elementi: basta il loro tag senza <>, separati da “,”; se solo spazio si intendono discendenti/nested; con + si può legare un tag hx con il suo successivo, adiacente p (adjacent sibling coordinator); combinabili a piacimento;

#id - seleziona ciò che in HTML ha il tag id (univoci per pagina);

.class - seleziona ciò che in HTML ha l’attributo class=””; anche elemento.class (solo elemento con quella classe), se applicato come elemento.class, non si può più usare solo .class per la categoria;

\*: selettore generico che prende tutto;

img[src] - seleziona solo src in img;

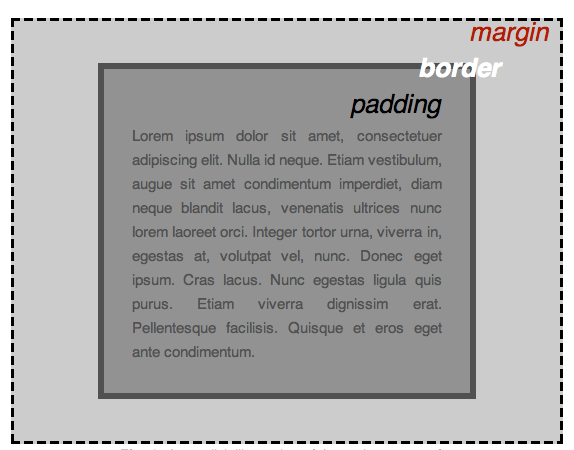
a:hover - per modificare il layout in uno specifico stato (in questo caso hovering); :visited; :link (unvisited)

<link href="<https://fonts.googleapis.com/css?family=Open+Sans>" rel="stylesheet" /> - add font;

px: pixels;

valori possono anche essere funzioni, come calc() per eseguire semplici operazioni; linear-gradient(direzione, colore1, colore2);

transform: prende varie funzioni, come rotate(), translate(), skew(),



CSS interpreta contenuti HTML come scatole impilate, dandogli varie proprietà, come padding, border e margini. nel Flexbox, si possono rendere le scatole flessibili.

Teoria del colore: {rgb(a), #000000, nome, hsl(a)%); a= trasparenza; hsl = hue (colore in deg), saturation (quanto è denso), luminosity (quanto è chiaro);

<https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Colors/Color_picker_tool> per scegliere il colore dalla palette e annotarsi i parametri;

proprietà che accettano nella stessa riga più elementi:

font, background, padding: (2 valori: up-down + L-R; 4 valori: up+R+down+L; auto=divide equamente); margin, border: (width, transparency, color);

display, width, font, text-shadow: (4 valori: horizontal offset, vertical offset, blur, color);

immagini sono elementi inline, pertanto per dare loro proprietà, aggiungere display: block;

@Rules: regole che css segue; come @import per importare altri files; @media per dare media queries, ossia condizionalità (come in HTML size);

Come browser legge HTML-CSS:

prima carica intero html, lo converte in memoria in dom (document object model) e poi esegue le richieste, come caricare immagini, css… JavaScript più tardi; spezzetta css e lo applica al giusto html; ogni tag di html diventa nodo, al quale sono applicati css e scripts; se browser non capisce regola css, o perché scritta male o non supportata, la ignora e passa avanti.

**JavaScript**

// comment here + invio; /\* comment here \*/

in HTML: <script src="scripts/main.js" defer></script> (defer fa caricare JavaScript più tardi di HTML);

Posizionarlo verso la fine, in modo che il sito carichi prima HTML.

querySelector(h1): fa riferimento a qualcosa in HTML

Javascript is case sensitive - Preferred to use camelCase;

var = global and local

let = local, block

const = block

block statement = azione di if, for…

in object: “” e simboli si richiamano con []

numeri interi:

no 0 = base 10;

0o = base 8;

ox = base 16;

ob = base 2;

` = template literal

\prima di carattere speciale per rappresentarlo\

Strings = “aa”; boolean: true/false; number: 1;

array: []; per riferirsi ad elemento: nomVar[0,1,2];

object: tutto {xx: “abc”;}

Operatori: +: somma tutto; -\*/: solo matematica; =: assegna valore; ===: compara valore e restituisce vero/falso se uguale (stretta uguaglianza); !, !==: non uguale, restituisce il contrario di vero/falso;

IF clause:

if(condizione da verificare) {conseguenza se vera+:} else {conseguenza se falsa+;}

Funzioni: introdotte da function, oppure standard come **alert()**; function(argomento, argomento, …) {cosa fa; definire variabili locali; return qualcosa;}. Per richiamarla, basta invocare il nome della funzione().

**addEventListener(evento scatenante, cosa fa);**

funzioni senza nome possono essere ridotte così: function (a,b) {...;} = (a,b) => {...;}

.getAttribute(attributo da ottenere); .setAttribute(attributo da modificare, cosa modificare);

**prompt():** mostra finestra di dialogo;