HTML - introduzione

HTML5 proviene da una specie di rivolta contro la strada che aveva intrapreso lo standard XHTML.

Da parte del gruppo di proponenti di HTML5 si voleva una definizione più flessibile dello standard, e la possibilità di aggiungere senza ostacoli nuove caratteristiche nel linguaggio.

Poi HTML5 si è affermato e oggi è dotato di una definizione tra gli standard del W3C, in collaborazione (forzata 😈) con WHATWG.

Anche il nome e` ormai denumeralizzato.

https://www.w3.org/html/
https://html.spec.whatwg.org/multipage/
https://it.wikipedia.org/wiki/HTML5

remember

Un'altra storia:

-2004: W3C announces XHTML2.0 – no backward compatibility; forget HTML, and XHTML

-Web Hypertext Application Technology WorkGroup says WHATwg!? *Vogliamo supporto ad HTML, meno rigore, backward compatibility, e non vogliamo chiudere *

(non e` proprio cosi` ☺)

-2007-2009: W3C cancella il progetto XHTML2.0

-HTML5 arises

- Compatibile con XHTML
- Meno rigoroso e disciplinato ma per documenti piu` interattivi (device capabilities; video and animations; graphics; editing, style, typography, and other)
- <nav>, <footer>, ... <aside> per layout
- <video>, <audio>, <canvas>, no prologue, doctype simple, no , drag&drop

HTML - introduzione

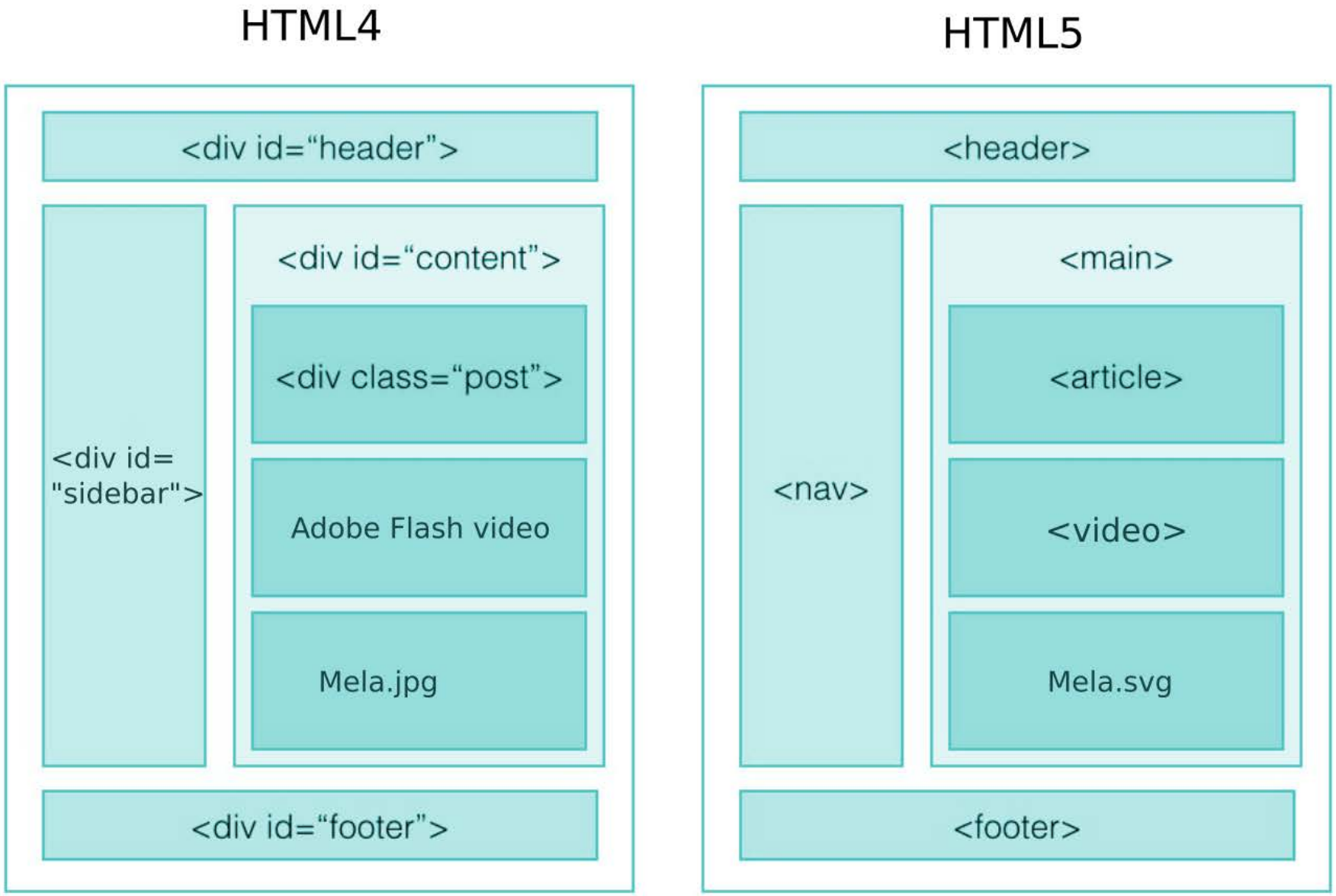
Negli anni 2000, il World Wide Web Consortium (W3C) ha sviluppato XHTML come evoluzione di HTML 4.01, introducendo una sintassi più rigorosa basata su XML. Tuttavia, questa rigidità ha reso lo sviluppo web più complesso e meno flessibile. In risposta, nel 2004, un gruppo di sviluppatori provenienti da Apple, Mozilla Foundation e Opera Software ha fondato la Web Hypertext Application Technology Working Group (WHATWG), con l'obiettivo di creare una nuova versione di HTML più flessibile e adattabile alle esigenze moderne del web.

Questo gruppo ha iniziato a lavorare su quella che sarebbe diventata HTML5, introducendo nuove funzionalità per applicazioni web più interattive e multimediali. Nel 2007, il W3C ha riconosciuto l'importanza di questo lavoro e ha iniziato a collaborare con la WHATWG per standardizzare HTML5.
La prima versione stabile di HTML5 è stata pubblicata come Raccomandazione dal W3C il 28 ottobre 2014.

Il risultato ha aspetti positivi e negativi. Da un punto di vista formale, la maggiore flessibilita` si e` anche tradotta in meno rigore del codice e piu` necessita` di lavoro da parte dei parser e dei motori di rendering (case doesn't matter, quotation unmatched, non-terminated elements, not wf indispensability)

Oggi, HTML5 (HTML) è lo standard attuale per la struttura e il contenuto delle pagine web, riconosciuto e mantenuto dal W3C.
https://www.w3.org/html/
https://html.spec.whatwg.org/multipage/
https://it.wikipedia.org/wiki/HTML5

HTML – Struttura del documento (idea ...)



HTML – Struttura del documento

HTML-layout_base.html

L'organizzazione della pagina web in HTML è basata su una serie di nuovi elementi strutturali (nuovi rispetto a XHTML)

- che hanno un significato semantico "atteso" specifico,
- e che in sostanza derivano da e non possono fare a meno di una specifica CSS che ne stilizzi la presentazione.

Elementi Principali

<article>

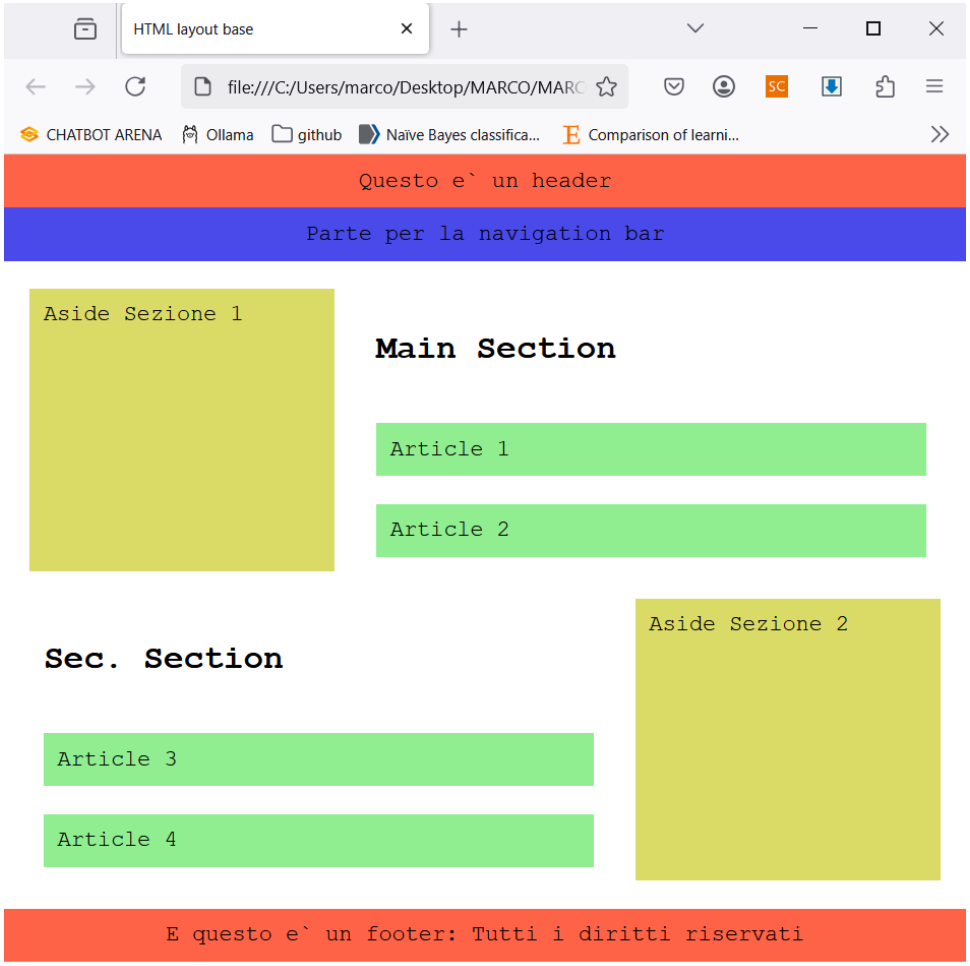
Contenuto autonomo e riutilizzabile (es. post, news).

<section>

Sezione logica di un documento.

<aside>

Contenuto marginale o complementare (es. barra laterale).



HTML – Struttura del documento

L'organizzazione della pagina web in HTML è basata su una serie di nuovi elementi strutturali (nuovi rispetto a XHTML)

- che hanno un significato semantico "atteso" specifico,
- e che in sostanza derivano da e non possono fare a meno di una specifica CSS che ne stilizzi la presentazione.

HTML5 ha introdotto nuovi elementi strutturali per sostituire l'uso generico di con tag dal significato semantico preciso. Tuttavia, questi elementi non modificano di per sé l'aspetto della pagina: è necessario uno stile CSS per definirne l'impaginazione e il comportamento visivo.

<article>: Rappresenta un contenuto autonomo e riutilizzabile, come un articolo di un blog, una notizia o una scheda prodotto. Può contenere altri elementi come titoli, paragrafi e immagini.

<section>: Identifica una sezione logica del documento, spesso con un titolo proprio. È utile per organizzare il contenuto in blocchi tematici.

<aside>: Contiene informazioni accessorie rispetto al contenuto principale, come barre laterali, citazioni o approfondimenti.

<main> e` destinato a contenere la parte informativa "dominante" del <body"; non puo` essere discendente di <article> et al.

La SEO (Search Engine Optimization) è l'insieme delle tecniche utilizzate per ottimizzare un sito web in modo che sia meglio indicizzato e classificato dai motori di ricerca (come Google). L'obiettivo è migliorare la visibilità del sito nei risultati di ricerca, aumentando il traffico organico (cioè non a pagamento).

Gli elementi visti qui sopra, con gli altri strutturali, e in particolare <main>, possono migliorare l'accessibilità e la SEO, fornendo agl screen reader (o alle device alternative di presentazione) ed ai motori di ricerca una struttura chiara del contenuto, facilitando la comprensione e l'indicizzazione delle pagine web.

ulteriori elementi strutturali

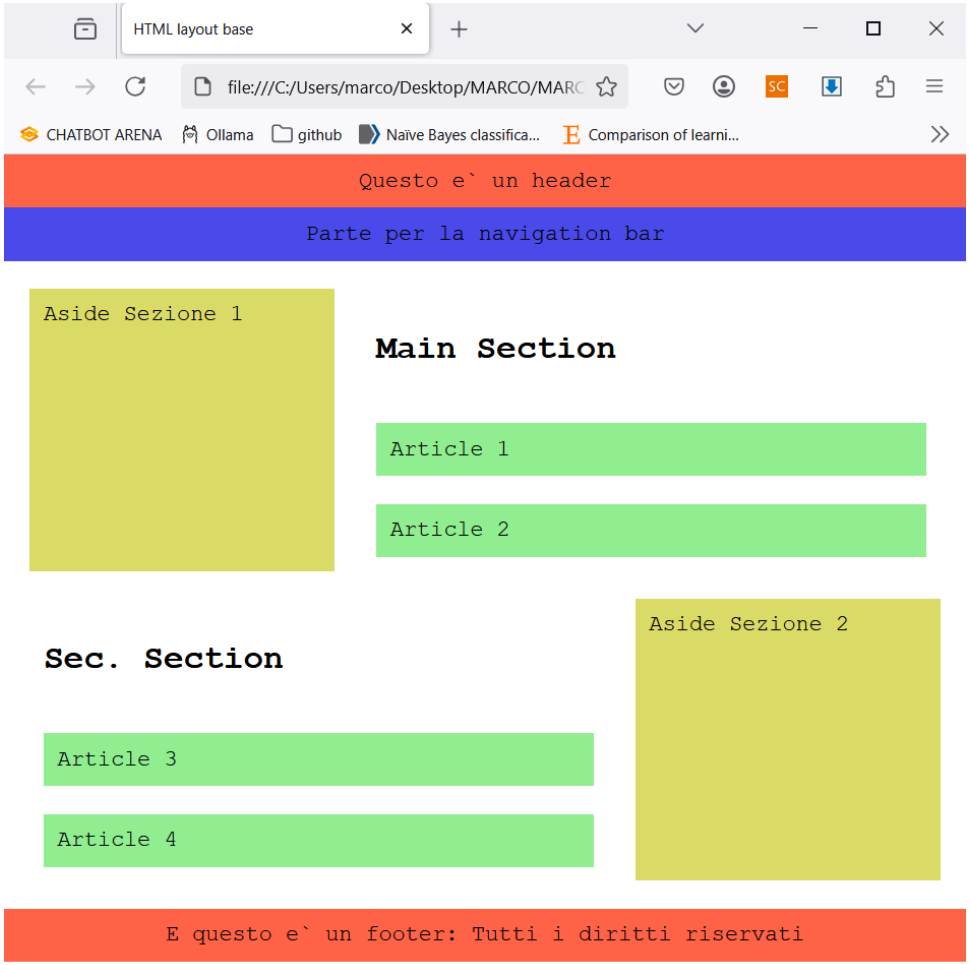
HTML-layout_base.html

O, come vengono anche chiamati da qualcuno "semantici".
In sostanza ci sia aspetta che vengano stilizzati in modo confacente al significato atteso.

<header>
Intestazione della pagina o di una sezione.

<footer>
Informazioni di chiusura o riferimenti.

<nav>
Contiene i link di navigazione.



ulteriori elementi strutturali

L'organizzazione della pagina web in HTML è basata su una serie di nuovi elementi strutturali (nuovi rispetto a XHTML)

- che hanno un significato semantico "atteso" specifico,
- e che in sostanza derivano da e non possono fare a meno di una specifica CSS che ne stilizzi la presentazione.

Oltre agli elementi <article>, <section> e <aside>, HTML5 introduce altri elementi strutturali che migliorano la leggibilità del codice e la SEO, rendendo la pagina più comprensibile sia per gli sviluppatori che per i motori di ricerca.

<header>: Rappresenta l'intestazione della pagina o di una sezione. Di solito contiene il logo, il titolo della pagina e i menu di navigazione. Può essere usato più volte in un documento HTML.

<footer>: Indica il piè di pagina, dove si trovano informazioni di chiusura, come copyright, contatti o link a risorse aggiuntive. Anche il <footer> può essere usato sia per l'intera pagina che per singole sezioni.

<nav>: Contiene i collegamenti principali per la navigazione all'interno del sito. È utile per indicare chiaramente ai motori di ricerca quali link sono rilevanti per la struttura del sito.

Questi elementi migliorano l'accessibilità e la separazione tra contenuto e struttura, rendendo il codice più leggibile e mantenibile.

Elementi per immagini e didascalie in HTML

L'uso di questi elementi è utile per migliorare la semantica del documento in generale, ed anche la sua accessibilita`:
a differenza di un semplice con un <p>, il <figcaption> è direttamente legato al contenuto multimediale, facilitando la comprensione sia per gli utenti che per i motori di ricerca.

<figure> Contiene immagini, grafici ...

<figcaption> Fornisce una didascalia descrittiva dell'immagine.

```
<html>
<head><title>...</title>

</head>

<body>
...
  <figure>
     <img src="./images/TomSawyer-
   HuckleberryFinn.png" alt="Tom and Huck">
     <figcaption>Illustration of Tom and
                              Huck </figcaption>
  </figure>
...
</body></html>
```

<figure> e <figcaption>

<figure> serve per contenere immagini, grafici, tabelle o altri contenuti multimediali, insieme alla loro didascalia.

L'elemento <figcaption> viene utilizzato per fornire una descrizione testuale associata all'elemento <figure>.

L'uso di questi elementi è particolarmente utile per migliorare la semantica del documento e l'accessibilità. Infatti, a differenza di un semplice con un <p>, il <figcaption> è direttamente legato al contenuto multimediale, facilitando la comprensione sia per gli utenti che per i motori di ricerca.

Esempio di utilizzo:

```
<figure>
  <img src="paesaggio.jpg" alt="Un paesaggio montano al tramonto">
  <figcaption>Vista di un tramonto sulle montagne.</figcaption>
</figure>
```

In questo esempio, il testo all'interno di <figcaption> descrive l'immagine, migliorando l'usabilità per gli utenti con disabilità visive che utilizzano screen reader.

altri elementi interessanti

...

vedi
meteo.html
in
3-additional_elements/

Campi <input> nelle form HTML e altro

Conosciamo già i principali campi (text, password, select, radio, checkbox, hidden, reset, submit) e adesso ne vediamo altri, introdotti in HTML5.

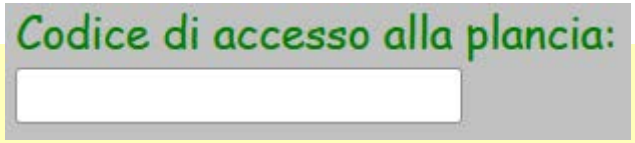
CAVEAT: la presentazione e la funzionalità dei campi possono variare a seconda del browser.

https://caniuse.com/ può aiutare a documentarsi sui comportamenti dei browser

Il browser, in generale, fornisce una validazione dei campi, gestita in modo però *browser dependent*.

NB

con <label> si può associare un'etichetta a un campo input, migliorando di molto la leggibilita` di una form, e sollevando il programmatore dall'incombenza di stilizzarla.




```
<html>
<head><title>...</title>
...
  <label for="password">Codice di accesso ...</label>
  <input type="password" id="password" name="password" required>
...
</body></html>
```

input-startrek.html

Campi <input> nelle form HTML e altro

L'elemento <input> è ben noto da prima, come uno dei piu` importanti, fondamentale per scambiare informazioni tra client e server, attraverso le form.

Presentazione e funzionalità possono variare a seconda del browser utilizzato. Ad esempio, alcuni browser offrono controlli avanzati per la selezione di date o colori, mentre altri forniscono un supporto più limitato.

In generale, i browser eseguono una prima validazione dei dati inseriti, in base al tipo di input specificato (ad esempio, un campo di tipo email richiede un formato valido di indirizzo email).

Rispetto a XHTML, HTML introduce nuovi tipi di campi <input> che esploreremo nelle prossime slide. Un concetto importante è l'uso di <label>, che permette di associare un testo descrittivo a un campo di input, migliorando l'accessibilità e l'usabilità.
Qui si usa un attributo id, nel campo input, per associare univocamente la label al campo: l'attributo for di <label> deve corrispondere all'id dell'elemento <input> associato.

Esempio:

```
<label for="fname">First name:</label>
<input type="text" id="fname" name="fname">
```

Cliccando sulla label, il browser attiva il campo di input corrispondente.

Diamoci una rinfrescata sui principali attributi dei campi input:

disabled: il campo è disabilitato

required: il campo è obbligatorio

placeholder: suggerisce un testo di esempio

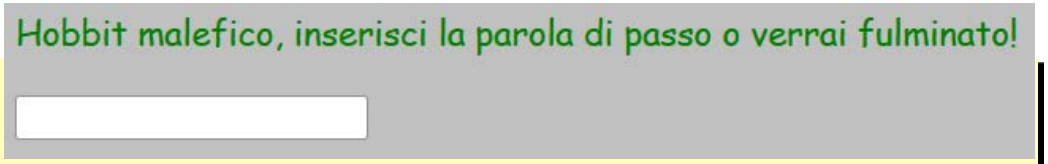
maxlength: lunghezza massima del valore inseribile

pattern: espressione regolare per la validazione

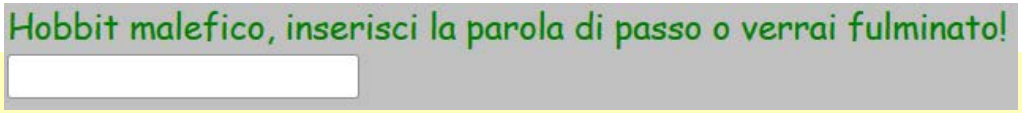
vedi dopo per altri ...

senza e con l'elemento <label>

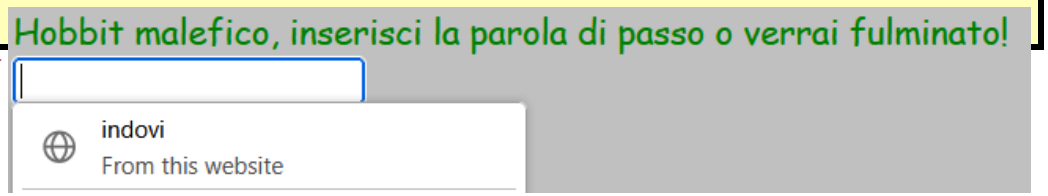
L'elemento <label> è usato nei form HTML per fornire un'etichetta descrittiva ai campi <input>, migliorando la chiarezza e l'accessibilità. Senza <label>, un campo può essere accompagnato solo da un semplice testo esterno, ma non sarà semanticamente collegato all'input.




```
<html>
<head><title>...</title>
...
  <p>Hobbit malefico, inserisci la parola di passo o verrai fulminato!</p>
  <input type="password" name="password">...
</body></html>
```




```
<html>
<head><title>...</title>
...
  <label for="pass">Hobbit malefico, inserisci la parola di passo o verrai fulminato!</label>
  <input type="password" id="pass" name="password">...
</body></html>
```



click sulla label

con e senza l'elemento <label> e altro

L’elemento <label> è usato nei form HTML per fornire un’etichetta descrittiva ai campi <input>, migliorando la chiarezza e l’accessibilità. Senza <label>, un campo può essere accompagnato solo da un semplice testo esterno, ma non sarà semanticamente collegato all’input.

Vediamo altri attributi che possiamo trovare dei campi input:

autocomplete: suggerisce valori basati su input precedenti.

readonly: il campo è visibile ma non modificabile.

size: specifica la larghezza del campo (in caratteri).

min / max: valori minimo e massimo (per numeri, date, ecc.).

step: determina l’incremento/decremento valido (utile per numeri e date).

Campi <input> avanzati

<input type="color"> permette di selezionare un colore.


```
<label for="favcolor">Scegli un colore: </label>
<input type="color" id="favcolor" name="favcolor">
```

<input type="number"> accetta solo numeri, con valori min/max e passi definiti.


```
<label for="eta">Inserisci la tua età:</label>
<input type="number" id="eta" name="eta" min="18" max="100"
step="1">
```

<input type="range"> crea uno slider, ma non mostra i valori.


```
<label for="volume">Regola il volume:</label>
<input type="range" id="volume" name="volume" min="0"
max="100" step="10">
```

Campi <input> avanzati (attributo placeholder)

l'attributo placeholder in un campo input serve a fornire un suggerimento sul contenuto o formato richiesto all'utente

```
<label for="name">Nome dell'ufficiale:</label>
<input type="text" id="name" name="name"
placeholder="esempio: Jean-Luc Picard" required>
```

input-startrek.html

Campi input avanzati, e attributo placeholder

HTML offre diversi tipi di campi <input> avanzati, che migliorano l'esperienza utente grazie ai controlli interattivi, che sono previsti più o meno in tutti i browser più diffusi.

type="color"
Permette di scegliere un colore tramite un selettore.
È utile per applicazioni grafiche o personalizzazioni di interfaccia.

```
<label for="favcolor">Scegli un colore:</label>
<input type="color" id="favcolor" name="favcolor">
```

type="number"
Accetta solo valori numerici e permette di impostare limiti con min, max e step. È utile per inserire dati quantitativi come età, quantità di prodotti, ecc.

```
<label for="eta">Inserisci la tua età:</label>
<input type="number" id="eta" name="eta" min="18" max="100" step="1">
```

type="range"
Mostra uno *slider* che l'utente può trascinare, ma non mostra direttamente il valore selezionato. È utile per impostare livelli come volume o luminosità.

```
<label for="volume">Regola il volume:</label>
<input type="range" id="volume" name="volume" min="0" max="100" step="10">
```

L'attributo placeholder ha usi abbastanza estesi; viene usato per suggerire un valore predefinito prima che l'utente digiti qualcosa.
È molto utile per migliorare la comprensibilità ed usabilità delle form. Non usatelo con type="color", range e number.

```
<label for="username">Nome utente:</label>
<input type="text" id="username" name="username" placeholder="Inserisci il tuo nome">
```

Altri campi <input> avanzati

<input type="search"> Campo per l'inserimento di testi di ricerca, con funzionalità simili a text, ma ottimizzato per le ricerche nei browser.

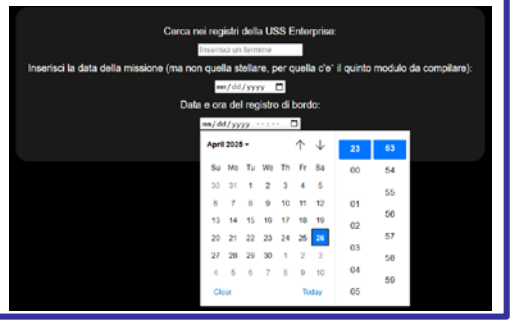
```
<label for="q">Cerca nella banca dati della Flotta:</label>
<input type="search" id="q" name="query" placeholder="Inserisci un termine...">
```

<input type="date"> Permette di selezionare una data tramite un calendario a comparsa (browser dipendente).

```
<label for="stardate">Inserisci la data stellare:</label>
<input type="date" id="stardate" name="stardate">
```

<input type="datetime-local"> Simile a date, ma permette di selezionare anche l'ora.

```
<label for="logtime">Registrazione del registro di bordo:</label>
<input type="datetime-local" id="logtime" name="logtime">
```



Altri campi input avanzati

I campi di input search, date e datetime-local forniscono funzionalità avanzate per l'inserimento di dati.

Il tipo search è pensato per i campi di ricerca, spesso fornendo un'icona di cancellazione automatica nei browser moderni. Questo aiuta l'utente a rimuovere rapidamente il testo inserito.

Il tipo date permette di selezionare una data da un calendario interattivo, senza dover digitare il formato manualmente. Tuttavia, il comportamento può variare tra i browser, e alcuni meno recenti potrebbero non supportarlo.

Il tipo datetime-local estende date, permettendo di selezionare anche l'orario. Questo è utile per applicazioni che richiedono timestamp precisi, come i log di sistema o le prenotazioni.

Altri tre campi <input> avanzati

`<input type="email">` accetta solo indirizzi e-mail validi.

```
<label>La tua email:</label>
    <input type="email" required>
```

La tua email:

`<input type="tel">` input per numeri di telefono, con possibilità di

```
<label for="tel">Il tuo numero di telefono:</label>
<input type="tel" id="tel" name="tel" pattern="\+?[0-9]{1,3}-[0-9]{10}"
                                          placeholder="+39-3421234567">
```

Il tuo numero di telefono:
+39-3421234567

`<input type="url">` accetta solo URL valide.

Altri tre campi input avanzati

Sono campi per i quali di solito e` offerta una validazione automatica dai browser.

email: accetta solo stringhe nel formato di un indirizzo e-mail valido (es. nome@example.com). I browser possono mostrare avvisi se il formato non è corretto.

tel: permette di inserire numeri di telefono.
Non ha una validazione predefinita, ma possiamo definirne una con l'attributo pattern.
Ad esempio, pattern="[0-9]{3}-[0-9]{3}-[0-9]{4}" accetta solo numeri nel formato 123-456-7890.

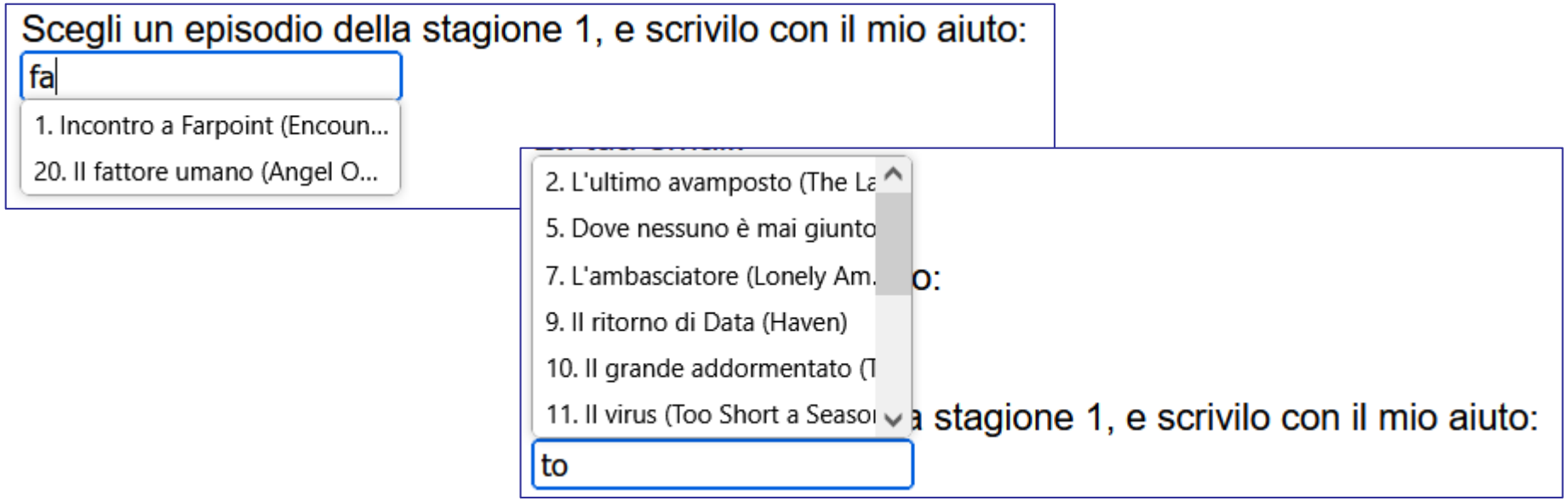
URL (url): richiede un URL valido (es. https://example.com).
Alcuni browser aggiungono automaticamente http:// se l'utente inserisce solo example.com.

Comunque,
la validazione più sicura, va fatta attraverso uno script, che spesso permette adattamenti più precisi alle vere intenzioni del programmatore (cioè ai requisiti fondamentali dell'applicazione web).
In altre parole ...

- lato server, ad esempio con uno script cgi o php o altro linguaggio,
- oppure lato client con javascript.

<input list ...>, con datalist


```
<label for="episodio">Scegli un episodio della stagione
1, e scrivilo con il mio aiuto:</label>
<input list="episodi" id="episodio" name="episodio">
<datalist id="episodi">
  <option value="1. Incontro a Farpoint">
  <option value="2. L'ultimo avamposto">
  <option value="3. Codice d'onore (Code of Honor)">
  ...
  <option value="26. Il segreto di Mordan IV">
</datalist>
```



<input list ...>, con datalist

Il campo list in HTML, utilizzato in combinazione con il tag <datalist>,
permette di creare un elenco di opzioni predefinite per un campo di input senza limitare l'utente a scegliere solo una di queste opzioni.
In altre parole, l'utente può digitare un valore libero nel campo di input, ma riceve delle suggerimenti o opzioni per completare il valore in base a quanto digitato, simile al comportamento di un'autocompletamento.

Il tag <datalist> consente di associare una lista di opzioni suggerite a un campo di input (<input>). L'utente può digitare un valore o selezionare uno dei suggerimenti proposti.

Ma e` una select?
No.
<select>: crea un elenco a discesa in cui l'utente può scegliere solo una delle opzioni predefinite. Non permette l'inserimento di valori liberi.
<datalist>: consente agli utenti di scrivere un valore personalizzato, ma offre anche delle opzioni suggerite in base ai valori presenti nel <datalist>.

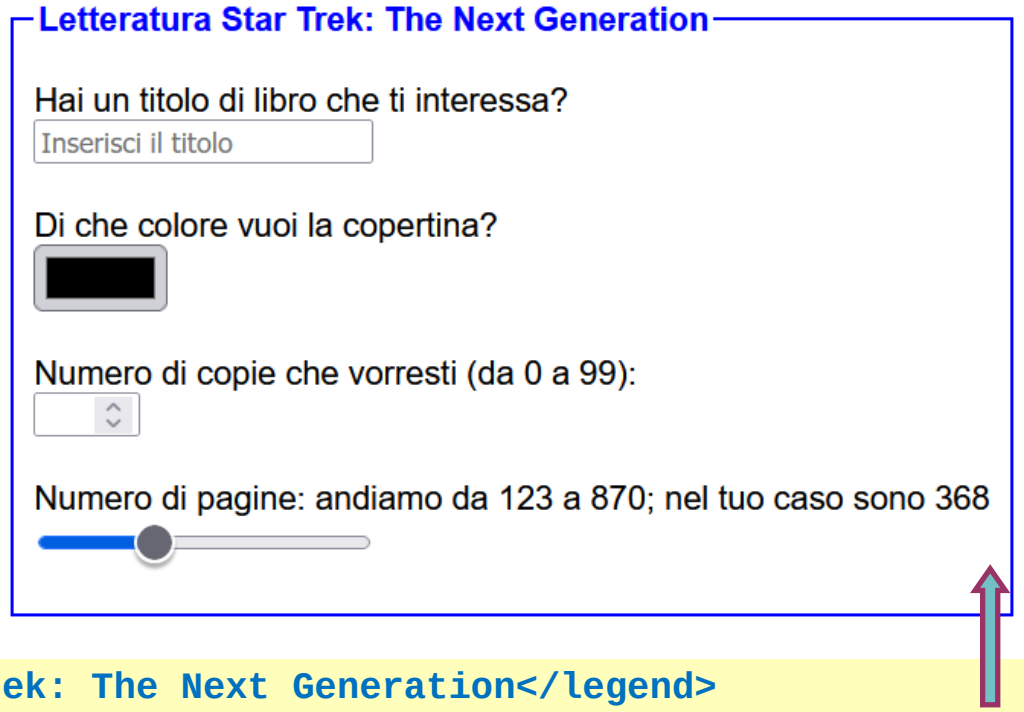
<fieldset> e <legend>

summa.html

<fieldset> è un elemento usato per raggruppare elementi correlati in una form.

<legend> fornisce una descrizione per il gruppo di elementi in un <fieldset>.

Annotano semanticamente parti della form migliorandone la comprensibilità e accessibilità.




```
<fieldset> <legend>Letteratura Star Trek: The Next Generation</legend>
<label for="book">Hai un titolo di libro che ti interessa?</label>
   <input type="text" id="book" name="book" placeholder="Inserisci il titolo">

<label for="color">Di che colore vuoi la copertina?</label>
   <input type="color" id="color" name="color">

<label for="quantity">Numero di copie che vorresti (da 0 a 99):</label>
   <input type="number" id="quantity" name="quantity" min="0" max="99">

<label for="pages">Numero di pagine: andiamo da 123 a 870; nel tuo caso sono
368</label>
   <input type="range" id="pages" name="pages" min="123" max="870" value="368">
</fieldset>
```

<fieldset> e <legend>

<fieldset> e <legend> vengono utilizzati insieme per organizzare e rendere più leggibili le form, soprattutto quando si gestiscono moduli complessi con molteplici campi.

Il tag <fieldset> crea un'area delimitata, che racchiude un gruppo di campi correlati all'interno di un modulo. Ad esempio, un modulo di registrazione potrebbe utilizzare un <fieldset> per separare la sezione dedicata ai dati anagrafici da quella relativa alla scelta di opzioni o preferenze.

<legend> fornisce un'etichetta descrittiva per il contenuto racchiuso all'interno del <fieldset>.

Iinsieme, questi elementi rendono un servizio utile all'accessibilità, della form e della pagina, e possono migliorare la comprensione di quel che c'è da immettere nella form da parte dell'utente.

Ricordiamoci che la presentazione della form potrebbe avvenire, ad esempio con un lettore vocale: in casi come questo aver aggiunto un po' di semantica (il raggruppamento di alcuni campi sotto la medesima etichetta esplicativa) permette un'esperienza d'uso migliore.

L'uso di <fieldset> e <legend> non solo migliora la semantica del documento HTML, ma aiuta anche nella gestione di moduli più grandi e complessi, dando a ciascun gruppo di campi una descrizione chiara e distinguibile.

La distinzione di un field set da un altro permette anche di sbizzarrirsi con CSS, con regole di presentazione dedicate solo a determinate sezioni del form.

Embedding (1/3)

Tre elementi permettono di inserire risorse esterne in un documento html: **<embed>**, **<iframe>**, **<object>**.

<embed>

Permette di inserire nel documento risorse multimediali, come immagini, video, audio, file pdf, contenuti interattivi come Flash o java applet, widget.

Tipicamente però immagini, video e audio sono serviti già da elementi più efficienti ...

Niente contenuto testuale, niente closing tag.
Attributi: **'src', 'type', 'width', 'height'...**

```
<h3>(embed di una risorsa pdf)</h3>
    <embed src="./resources/lwebCharacters.pdf"
           type="application/pdf" width="500" height="400">

<h3>(embed di una risorsa html non interattiva)</h3>
    <embed type="text/html" src="./resources/quadrati-css.html"
           style=" width: 100vw; height: 100vh;">
```

Embedding (2/3)

Tre elementi permettono di inserire risorse esterne in un documento html: **<embed>**, **<iframe>**, **<object>**.

<iframe>

Usato per inserire un altro documento html nella pagina.

In sostanza mostra una pagina separata, in un'area delimitata della pagina corrente.

Permette di inserire contenuti esterni, anche interattivi come mappe.

Attributi: **`src`, 'width', 'height, 'allowfullscreen' ...**

```
<h3>(stessa cosa ma con iframe)</h3>
    <iframe src="./resources/quadrati-css.html"
            style="width: 100%; height: 100vh"></iframe>

<h3>(iframe della pagina in cui il contenuto cambia al tocco di una falange></h3>
    <iframe src="./resources/jsModificaPageContent.html" style="width: 100%; height: 50wh"></iframe>
```

Embedding (2/3)

Una possibilità di **<iframe>**, che manca agli altri metodi di *embedding* è il *sandboxing*, che permette di eseguire l'eventuale codice che arriva con la risorsa, in un modalità ristretta e controllata.
L'attributo è **sandbox**.

```
<iframe src="./resources/curling.mp4" width="25%"
sandbox="allow-scripts allow-same-origin">
    Your browser does not support iframes or this
content.
</iframe>
```

allow-scripts = permette l'esecuzione di script;

allow-same-origin = permette al contenuto inserito di scambiare informazioni con il suo server di origine (forms e cookie sono bloccati)

Embedding (3/3)

Tre elementi permettono di inserire risorse esterne in un documento html: **<embed>**, **<iframe>**, **<object>**.

<object>

permette di "embeddare" cose come quelle viste nelle slide precedenti, ma ha qualche aspetto di flessibilità per cui vale la pena di parlarne un po'.

Attributi: **'data', 'type', 'height', 'width', 'name' ...**

<object> permette operazioni di *fallback*, cioè operazioni eseguite quando qualcosa non va, in alternativa a quel che non si può fare.

Ne abbiamo visto un esempio nella slide precedente: se l'iframe o il contenuto dell'iframe non e` supportato dal browser, viene stampato un messaggio.

<object> è più flessibile, nel senso che, oltre al semplice messaggio testuale scritto prima del tag di chiusura, permette di aggiungere link e anche chiamare l'esecuzione di script, in conseguenza del problema

```
<object type="text/html" data="... " style="...">
   <p style="color: red; ... border: solid blue thick;">
                    Your browser does not support embedded objects.</p>
   <img src="..." alt="Error image" style="margin: 10px; width: ... ">
   <p>Click <a href="..." style="...">here</a> for information.</p>
</object>
```

Frameworks

Per scrivere CSS usiamo **"Vanilla CSS"**

quello che conosciamo

scrittura manuale di regole di stile: flessibile ma più laboriosa

Esistono i framework che forniscono aiuto nella stilizzazione, mediante classi e componenti predefinite, e permettono di CCS-are meno

Tailwind CSS

approccio *Utility First:* classi piccole e riutilizzabili direttamente nell'HTML

```
<button class="px-4 py-2 bg-blue-500 text-white
                rounded">Click Me</button>
```

Bootstrap

approccio *Component-based*:

set di componenti predefiniti pronti all'uso

```
<button class="btn btn-primary">Click Me</button>
```

Obiettivo: velocizzare lo sviluppo, garantire coerenza e responsività

Per scrivere CSS usiamo **"Vanilla CSS"**

quello che conosciamo

scrittura manuale di regole di stile: flessibile ma più laboriosa

```
.custom-button {
  padding: 0.5rem 1rem;       /* Tailwind px-4 (hor. padding)
                                 and py-2 (vert. padding) */
  background-color: #3b82f6;  /* bg-blue-500 */
  color: white;               /* text-white */
  border: none;
  border-radius: 0.25rem;     /* Tailwind rounded */
  cursor: pointer;            /* manina */
}

.custom-button:hover {
  background-color: #2563eb;
}

...

<button class="custom-button">Click Me</button>
```

Frameworks

Quando si lavora allo stile di una pagina web, si può procedere scrivendo direttamente codice CSS, pratica conosciuta come Vanilla CSS.

Questo approccio fornisce massima libertà al programmatore, che definisce selettori, regole e strutture secondo le proprie esigenze. Pero`, in progetti complessi o di grandi dimensioni, mantenere ordine e coerenza nel codice CSS può diventare complicato e dispendioso in termini di tempo. Inoltre, la gestione della responsività, del riutilizzo del codice e della compatibilità tra browser può richiedere uno sforzo significativo.

Niente vero ... soprattutto non ci si vuole lambiccare il cervello per riuscire a scrivere il css necessario a realizzare proprio quello che vogliamo, oppure si vuole fare un lavoro più veloce (anche a prezzo di non ottenere proprio la cosa bella che si aveva in testa.

I **framework** CSS rispondono a questa esigenza di semplificazione e velocizzazione del processo di sviluppo.

Questi strumenti offrono soluzioni pronte per l'uso, riducono la necessità di scrivere codice da zero e promuovono uno stile visivo coerente tra le pagine.

In realtà, si tratta di strumenti fondamentali per lo sviluppo web moderno, in grado di potenziare l'efficienza, migliorare la manutenibilità del codice e assicurare una buona esperienza utente.

Due dei framework più diffusi e con approcci profondamente diversi sono Tailwind CSS e Bootstrap.

- Tailwind CSS funziona con un approccio chiamato "Utility First" (low-level utility classes).
- Bootstrap, invece, è un framework basato su componenti pronte per l'uso (e anche classi, ma preparate per essere presentate secondo il design di Bootstrap) .

Frameworks

Due dei framework più diffusi e con approcci profondamente diversi sono Tailwind CSS e Bootstrap.

- Tailwind CSS funziona con un approccio chiamato "Utility First". In pratica, ti dà un sacco di classi CSS già pronte, ognuna fa una cosa precisa: ad esempio, p-4 aggiunge spazio (padding), bg-blue-500 mette uno sfondo blu. Usi queste classi direttamente nell'HTML, così puoi vedere subito cosa fa ogni parte del codice.

  Il bello di Tailwind è che puoi costruire velocemente e personalizzare tutto come vuoi, senza dover scrivere mille righe di CSS a parte. Il tuo sito avrà uno stile tutto tuo, ma comunque ordinato e coerente.
  L'unico problema? L'HTML può diventare un po' pieno di classi e magari più difficile da leggere per chi non conosce Tailwind.

https://tailwindcss.com/docs/customizing-colors#default-color-palette
https://tailwindcss.com/docs/font-size
https://tailwindcss.com/docs/padding — spacing
https://tailwindcss.com/docs/flex — flexbox utilities
https://nerdcave.com/tailwind-cheat-sheet
https://tailwindcss.com/docs/ — usare il menu` a sinistra

- Bootstrap, invece, ...

Frameworks

Due dei framework più diffusi e con approcci profondamente diversi sono Tailwind CSS e Bootstrap.

- Tailwind CSS funziona con un approccio chiamato "Utility First". In pratica, ti dà un sacco di classi CSS già pronte, ognuna fa una cosa precisa: ad esempio, p-4 aggiunge spazio (padding), bg-blue-500 mette uno sfondo blu. Usi queste classi direttamente nell'HTML, così puoi vedere subito cosa fa ogni parte del codice.

  Il bello di Tailwind è che puoi costruire velocemente e personalizzare tutto come vuoi, senza dover scrivere mille righe di CSS a parte. Il tuo sito avrà uno stile tutto tuo, ma comunque ordinato e coerente.
  L'unico problema? L'HTML può diventare un po' pieno di classi e magari più difficile da leggere per chi non conosce Tailwind.

- Bootstrap, invece, è un framework basato su componenti riutilizzabili. Una cassetta degli attrezzi piena di pezzi già pronti per costruire siti web. Ha un sacco di cose utili come pulsanti, finestre modali, griglie, menu e form, tutti già fatti e belli da vedere. Ogni pezzo ha uno stile preimpostato, moderno e che si adatta bene a schermi grandi e piccoli (responsive).

  Bootstrap è perfetto se vuoi fare siti belli e ordinati senza essere un esperto di CSS.
  Ti fa risparmiare tempo e ti aiuta a creare pagine che funzionano bene e sono facili da usare.
  Però fai attenzione: se usi tutto così com'è, il tuo sito potrebbe sembrare uguale a tanti altri, con quel classico "aspetto Bootstrap" che si riconosce subito.
  Se vuoi che il tuo sito sia unico, dovrai personalizzare un po' i componenti.

https://getbootstrap.com/docs/5.3/getting-started/introduction/
https://www.w3schools.com/bootstrap5/
https://www.html.it/guide/guida-bootstrap/

Esempio Tailwind (1/4)

Personaggi Letterari 3 riscritto con Tailwind

Alcuni elementi ora sono stilizzati usando le classi TailwindCSS

Body:

```
<body class="bg-gray-100 text-gray-800 flex flex-col min-h-
screen font-serif leading-relaxed">
```

bg-gray-100:	imposta lo sfondo a grigio chiaro **(#f5f5f5).**
text-gray-800:	testo di colore grigio scuro.
flex flex-col:	layout a colonna con Flexbox.
min-h-screen:	altezza minima pari all'intera altezza della finestra.
font-serif:	vogliamo la font con le grazie.
leading-relaxed:	interlinea leggermente aumentata per leggibilità.

Header:

```
<header class="bg-orange-500 text-white text-center p-4">
```

bg-orange-500: **sfondo arancione medio.**	
text-white:	testo bianco.
text-center:	testo centrato.
p-4:	padding interno su tutti i lati (rem = root em).

Esempio Tailwind (2/4)

Link nell'elemento <nav>:

```
<a href="#tom-sawyer" class="mx-4 hover:underline">Tom Sawyer</a>
```

my-2 sarebbe 0.5rem sull'asse verticale ... top e bottom

mx-4: margine orizzontale (left e right) di 1 rem.
hover:underline: sottolinea il testo al passaggio del mouse.

Main:

```
<main class=" flex-1 m-4 bg-gray-300 p-4 rounded">
```

flex-1: occupa tutto lo spazio disponibile nel contenitore Flex.
m-4: margine ovunque di 1 rem.
bg-gray-300: sfondo grigio medio.
p-4: padding interno.
rounded: angoli leggermente arrotondati.

Esempio Tailwind (3/4)

Article:

```
<article class="mb-4 p-4 bg-gray-100 border border-gray-300 rounded">
```

mb-4: margine inferiore.
p-4: padding.
bg-gray-100: sfondo chiaro.
border border-gray-300: bordo sottile grigio chiaro.
rounded: angoli arrotondati.

Titolo di secondo livello:

```
<h2 class="text-xl font-semibold mb-2">Tom Sawyer & Huckleberry Finn</h2>
```

text-xl: dimensione testo extra-large.
font-semibold: grassetto medio.
mb-2: margine inferiore.

Esempio Tailwind (4/4)

Immagini:

```
<img src="./images/TomSawyer-HuckleberryFinn.png" class="inline-block w-1/3">
```

inline-block: visualizzato in linea ma trattato come blocco.
w-1/3: larghezza pari a un terzo del contenitore padre.

Aside:

```
<aside class="lg:w-1/4 bg-orange-200 p-4 m-4 border-l-4 border-orange-500 rounded">
```

lg:w-1/4: larghezza 25% a partire da schermi grandi.
bg-orange-200: sfondo arancione chiaro.
p-4: padding.
m-4: margine ovunque di 1rem.
border-l-4: bordo sinistro spesso.
border-orange-500: colore del bordo arancione.
rounded: angoli arrotondati.

Link nell'Aside:

```
<a href="#" class="text-blue-600 hover:underline">
```

text-blue-600: colore del testo blu, medium dark.
hover:underline: sottolineato al passaggio del mouse.

Esempio Bootstrap (1/4)

Personaggi Letterari 3 riscritto con Bootstrap

All'inizio bisogna caricare la libreria apposita (a meno di non avere tutto sul proprio server).

```
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.
3.0/dist/css/bootstrap.min.css"
rel="stylesheet">
```

la versione *minified* della libreria bootstrap viene caricata dal content delivery system jsDelivr; questa sara` la stylesheet principale di questa pagina

E poi gli elementi vengono ri-stilizzati

Body:

```
<body class="bg-light text-dark d-flex flex-column min-vh-100">
```

bg-light: sfondo grigio chiaro (BS light gray background).

text-dark: testo di colore scuro.

d-flex flex-column: layout flexbox in colonna.

min-vh-100: altezza min 100% dell'altezza della finestra.

Esempio Bootstrap (2/4)

Header:

```
<header class="bg-warning text-white text-center p-3">
```

bg-warning: colore di sfondo arancione (giallo-arancio standard BS). [modificato]

text-white: testo bianco.
text-center: allinea il testo al centro.
p-3: padding su tutti i lati (3 = 1rem).

Link di Navigazione:

```
<a href="#tom-sawyer" class="mx-3 text-decoration-none text-dark">
```

mx-3: margine sulla linea orizzontale 1rem.
text-decoration-none: rimuove la sottolineatura.
text-dark: testo scuro.

Esempio BS (3/4)

Article:

```
<article class="mb-3 p-3 bg-white border rounded shadow-sm">
```

mb-3: margine bottom 1rem.
p-3: padding interno.
bg-white: sfondo bianco.
border: bordo grigio chiaro.
rounded: angoli arrotondati.
shadow-sm: ombricina per profondità.

img:

```
<img src="..." class="img-fluid w-50">
```

img-fluid: immagine responsive (max-width: 100% e height: auto).
w-50: larghezza 50% del contenitore.

figcaption:

```
<figcaption class="fst-italic mt-2 small">
```

fst-italic: font stile italics.
mt-2: margine top di 0.5rem.
small: testo più piccolo del normale.

Esempio BS (4/4)

Aside:

```
<aside class="col-lg-3 bg-warning-subtle p-3 border-start
border-warning rounded ms-lg-3 mt-3 mt-lg-0">
```

col-lg-3: larghezza 3 colonne su 12 colonne.
bg-warning-subtle: versione chiara del colore "warning". [modificato]
p-3: padding interno.
border-start: bordo sinistro.
border-warning: colore del bordo: arancione. [modificato]
rounded: angoli arrotondati.
ms-lg-3: margine sinistro solo su schermi larghi.
mt-3 mt-lg-0: margine top su piccoli schermi, rimosso su grandi.

Footer:

```
<footer class="bg-warning text-white text-center p-3 mt-auto">
```

mt-auto: margin top auto (spinge il footer in basso).

Audio Player:

```
<audio controls class="d-block mx-auto">
```

d-block: trasforma l'elemento in blocco (per centratura).
mx-auto: margine orizzontale auto: centrato.

Attività in laboratorio

html 1

in 1-HTML-layout_base/

Consultare HTML-layout_base.html e veder come sono distribuiti gli elementi.

Impadronirsi della pagina inserendovi contenuti di proprio gusto.

html 2

in 1-HTML-layout_base/

Poi vedere, attraverso il browser, HTML-layout_base2.html ed eseguire l'esercizio che è descritto proprio lì (Dovrebbe venir fuori HTML-layout_base3.html ...)

html 3

in 2-HTML-structural-fig-audio/

html-struct-elements-0.html propone un primo modello di uso degli elementi strutturali; consultarlo e poi produrre un sito che sia basato su argomenti di proprio gusto.

html 4

in 2-HTML-structural-fig-audio/

html-struct-elements-1.html propone un layout un po` piu` carino.

Ci sono immagini, ma vanno sistemate. Troppo grandi.

Poi c'è un elenco di link che è posto in fondo, e invece starebbe bene a lato, in un elemento aside posto sulla destra, o sulla sinistra, della pagina.

Fare quanto sopra. E poi applicare quello che si e` fatto al sito creato nell'esercizio precedente

Una idea di soluzione è in html-struct-elements-2.html, dove sono inserite anche risorse multimediali, per ora solo audio.

Attività in laboratorio

html quinto

in 2-HTML-structural-fig-audio/

ZZZ-esercizio-flex.html propone una bella pagina.

I riquadri che rimandano ai dettagli però sono disallineati: usare flex per riallinearli.

Il testo dell'esercizio è nella pagina stessa.

ZZZ-esercizio-flex-sol.html è quel che si può immaginare.

html6 html7 html8

Usare l'esempio * per sperimentare quanto discusso nella slide di riferimento e nel file html, producendo uno o più documenti (quindi un sito web) in cui vengano usati, non tutti insieme nel medesimo file, gli elementi in esame. L'argomento sotteso al sito è a piacere. Un contenuto di queste pagine potrebbe essere la spiegazione degli elementi usati (meglio di come ha fatto il professore in meteo.html).

* = meteo.html, datalist.html, date.html, ...

html9

Aggiungere in uno dei file prodotti in precedenza, un esempio di <picture>, uno di <audio> ed uno di <video>, secondo i seguenti esempi.

```
<audio controls>
   <source src="./resources/A_Zonzo.mp3" type="audio/mpeg">
   Your browser does not support the audio element.
</audio>
<video controls>
  <source src="curling.mp4" type="video/mp4">
  Il tuo browser non supporta il video.
</video>
<picture>
  <source srcset="immagine-grande.webp" type="image/webp" media="(min-width: 800px)">
  <source srcset="immagine-piccola.jpg" type="image/jpeg" media="(max-width: 799px)">
  <img src="immagine-piccola.jpg" alt="Descrizione immagine">
</picture>
```

Attività in laboratorio

html10

in 7-esperienze/

consultare il file quadrati.css.html, in esperienze/quadrati-css/ per familiarizzarsi con l'uso della trasparenza, nel disegnare quadrati con css.

La funzione rgb() ha anche un quarto parametro che denota la trasparenza della colorazione, qui usato.

html11

in 5-embed/

consultare il file embed.html.

Familiarizzare con l'uso di <embed>, <iframe> e <object> sulla base degli esempi mostrati.

Modificando solo per lo stretto necessario il file, sperimentare cosa succede di diverso con object, rispetto a embed e iframe, quando la risorsa richiesta non è disponibile.

Sperimentare anche l'uso del codice commentato (che fornisce un diverso esempio di uso di object.

html12 (esercizio diabolico)

Considerare il proprio homework1.

Ristrutturarlo usando quanto visto nelle slide, con riferimento agli esempi in 2-HTML-structural-fig-audio/, 3-additional_elements/, 4-inputFields/.

Anche 5-embed ed esperienze/ potrebbero essere utili ...

html13 ***in 7-esperienze/***

Nella directory ci sono tre file che possono fornire qualche esperienza addizionale.

- un file html con l'uso della geolocalizzazione
- uno con la evocazione di una tastiera virtuale nel momento in cui si vuole compilare un campo testuale
- una esperienza d'uso di <canvas>, in due versioni, la prima delle quali è abbastabza semplice.