HTML

Storia e Evoluzione di HTML

Origini di HTML

HTML, acronimo di **HyperText Markup Language**, è stato sviluppato nei primi anni '90 da Tim Berners-Lee, un ingegnere informatico britannico. La sua creazione aveva l'obiettivo di facilitare la condivisione di informazioni attraverso il World Wide Web. La prima specifica di HTML, nota come HTML 1.0, fu pubblicata nel 1993.

Evoluzione di HTML

1. **HTML 1.0 (1993)**
    - Prima versione standard.
    - Forniva le basi per la creazione di pagine web statiche.
2. **HTML 2.0 (1995)**
    - Standardizzato dal W3C (World Wide Web Consortium).
    - Introdusse nuovi elementi e attributi per migliorare la formattazione e l'interattività.
3. **HTML 3.2 (1997)**
    - Aggiornamenti significativi, incluso il supporto per applet Java e script.
    - Introduzione di tabelle e nuovi tag per la formattazione.
4. **HTML 4.0 (1997)**
    - Maggiore focus sull'accessibilità e sulla separazione del contenuto dalla presentazione.
    - Introdotto il concetto di CSS (Cascading Style Sheets) per la gestione del layout.
5. **HTML 4.01 (1999)**
    - Aggiornamento della versione precedente con correzioni e miglioramenti minori.
6. **XHTML 1.0 (2000)**
    - Ristrutturazione di HTML in conformità con le regole XML.
    - Maggiore rigore nella sintassi.
7. **HTML5 (2014)**
    - Introduzione di nuove funzionalità per supportare multimedia e grafica (audio, video, canvas).
    - Miglioramenti per la semantica del markup e nuove API (come la geolocalizzazione).
    - Eliminazione di elementi obsoleti e supporto per applicazioni web moderne.
8. **HTML Living Standard (2014 - presente)**
    - HTML non è più standardizzato in versioni numerate.
    - Aggiornamenti continui attraverso il WHATWG (Web Hypertext Application Technology Working Group).
    - Introduzione di nuove funzionalità e miglioramenti in tempo reale.

Importanza di HTML

HTML è fondamentale per la creazione di pagine web e rappresenta la struttura di base di qualsiasi sito. La sua evoluzione ha permesso di adattarsi alle nuove tecnologie e alle esigenze degli sviluppatori, rendendo il web sempre più interattivo e accessibile.

Conclusione

L'evoluzione di HTML riflette i cambiamenti nella tecnologia web e le esigenze degli utenti. Con l'emergere di HTML Living Standard, il linguaggio continuerà a evolversi, assicurando che il web rimanga un ambiente dinamico e innovativo.

Struttura di Base di un Documento HTML

```
<!DOCTYPE html>
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Il Titolo della Pagina</title>
    <link rel="stylesheet" href="stile.css">
</head>
<body>
    <header>
        <h1>Benvenuto nel mio sito web</h1>
    </header>
    <nav>
        <ul>
            <li><a href="#home">Home</a></li>
            <li><a href="#about">Chi Siamo</a></li>
            <li><a href="#contact">Contatti</a></li>
        </ul>
    </nav>
    <main>
        <section id="home">
            <h2>Home</h2>
            <p>Questo è un paragrafo di esempio.</p>
        </section>
        <section id="about">
            <h2>Chi Siamo</h2>
            <p>Informazioni sul nostro progetto.</p>
        </section>
        <section id="contact">
            <h2>Contatti</h2>
            <p>Come raggiungerci.</p>
        </section>
    </main>
    <footer>
        <p>&copy; 2024 Il Tuo Nome</p>
    </footer>
</body>
</html>
```

Spiegazione degli Elementi

1. **<!DOCTYPE html>**:
    - Dichiara il tipo di documento e la versione di HTML utilizzata (in questo caso, HTML5).
2. **<html lang="it">**:
    - L'elemento radice del documento, con l'attributo lang che indica la lingua del contenuto (italiano).
3. **<head>**:
    - Contiene metadati e informazioni sulla pagina, come il titolo e i link a fogli di stile.
    - **<meta charset="UTF-8">**: specifica la codifica dei caratteri.
    - **<meta name="viewport" content="width=device-width, initial-scale=1.0">**: rende la pagina responsive sui dispositivi mobili.
    - **<title>**: il titolo della pagina, visibile nella scheda del browser.
    - **<link rel="stylesheet" href="stile.css">**: collega un foglio di stile CSS esterno.
4. **<body>**:
    - Contiene il contenuto visibile della pagina.
    - **<header>**: sezione introduttiva, spesso utilizzata per il titolo e la navigazione.
    - **<nav>**: contiene i link di navigazione del sito.
    - **<main>**: sezione principale del contenuto, suddivisa in **<section>**.
    - **<footer>**: sezione finale, tipicamente usata per informazioni legali o di contatto.

Spiegazione di meta charset="UTF-8"

L'elemento <meta charset="UTF-8"> è una parte fondamentale della sezione <head> di un documento HTML.

Charset (o "codifica dei caratteri") si riferisce al sistema utilizzato per rappresentare i caratteri in un documento. Ogni carattere (lettere, numeri, simboli) deve essere codificato in un modo che il computer possa comprendere.

UTF-8 (Unicode Transformation Format - 8-bit) è una delle codifiche più comuni e utilizzate nel web. È parte dello standard Unicode, che permette di rappresentare quasi tutti i caratteri di tutte le lingue scritte del mondo.

UTF-8 è particolarmente vantaggioso perché:

Compatibilità: È retro compatibile con ASCII, il che significa che i primi 128 caratteri di UTF-8 sono identici a quelli di ASCII.

Flessibilità: Può rappresentare caratteri da diversi alfabeti e simboli, rendendolo ideale per pagine web multilingue.

Efficienza: Utilizza una lunghezza variabile per la codifica dei caratteri; i caratteri ASCII occupano 1 byte, mentre i caratteri più complessi possono occupare fino a 4 byte.

Corretta Visualizzazione:

Se non si specifica il charset, il browser potrebbe assumere una codifica predefinita, che potrebbe non essere corretta. Ciò può portare a caratteri illeggibili o errati, specialmente per lingue non latine.

Cosa sono i form?

Immagina di compilare un modulo online per iscriverti a un corso o fare un acquisto. Ecco, quella struttura che ti permette di inserire i tuoi dati si chiama form. In HTML, i form sono utilizzati per raccogliere informazioni dagli utenti e inviarle a un server.

A cosa servono i form?

I form hanno moltissimi utilizzi, come:

- **Iscrizioni:** Per raccogliere dati come nome, email e password.
- **Contatti:** Per permettere agli utenti di inviarti messaggi.
- **Ricerche:** Per consentire agli utenti di cercare informazioni su un sito.
- **Acquisti:** Per raccogliere i dati di pagamento e spedizione.
- **Sondaggi:** Per raccogliere opinioni e feedback.

Come creare un form in HTML?

Per creare un form, utilizziamo il tag <form>. All'interno di questo tag, inseriamo gli elementi che compongono il form, come:

- **<input>:** Questo tag è il più comune e serve per creare vari tipi di campi di input, come caselle di testo, password, checkbox, radio button, ecc.
- **<label>:** Questo tag associa un'etichetta a un campo di input, rendendo più chiaro il suo scopo.
- **<button>:** Questo tag crea un pulsante che invia il form.

Esempio semplice:

HTML

```
<form>
  <label for="nome">Nome:</label>
  <input type="text" id="nome" name="nome">

  <label for="email">Email:</label>
  <input type="email" id="email" name="email">

  <button type="submit">Invia</button>

</form>
```

In questo esempio, abbiamo creato un form con due campi: uno per il nome e uno per l'email. Quando l'utente clicca sul pulsante "Invia", i dati inseriti nei campi verranno inviati a un server.

Cosa succede quando invio un form?

Quando invii un form, i dati che hai inserito vengono solitamente inviati a un file sul server (spesso un file PHP o Python). Questo file elabora i dati e può eseguire diverse azioni, come:

- **Inviare una email:** Ad esempio, se il form è un modulo di contatto.
- **Salvare i dati in un database:** Ad esempio, per gestire le iscrizioni a un sito.
- **Eseguire altre operazioni:** Come generare un PDF o visualizzare un messaggio di conferma.

Approfondiamo insieme i form HTML con gli elementi <select> e <textarea>! Sono strumenti molto utili per creare moduli interattivi e raccogliere diverse tipologie di informazioni dagli utenti.

<select>: Scegliere tra diverse opzioni

L'elemento <select> ti permette di creare una lista a discesa (o un menu a tendina) da cui l'utente può selezionare una sola opzione. È perfetto per offrire delle scelte predefinite, come ad esempio:

- **Un elenco di paesi:**

```
<select name="paese">
  <option value="Italia">Italia</option>
  <option value="Francia">Francia</option>
  <option value="Spagna">Spagna</option>
</select>
```

- **Le tag di un articolo:**

```
<select name="tag">
  <option value="tecnologia">Tecnologia</option>
  <option value="viaggi">Viaggi</option>
  <option value="cucina">Cucina</option>
</select>
```

Attributi utili:

- **multiple:** Permette all'utente di selezionare più opzioni contemporaneamente.
- **size:** Definisce il numero di opzioni visibili senza aprire la lista a discesa.
- **selected:** Indica l'opzione selezionata di default.
-

<textarea>: Scrivere testi più lunghi

L'elemento <textarea> è ideale per raccogliere testi più lunghi, come commenti, messaggi o descrizioni. È come una piccola area di testo dove l'utente può scrivere liberamente

```
<textarea name="messaggio" rows="5" cols="30" placeholder="Scrivi qui il tuo messaggio..."></textarea>
```

- **rows:** Definisce il numero di righe visibili.
- **cols:** Definisce il numero di colonne visibili.
- **placeholder:** Mostra un testo di esempio all'interno dell'area, che scompare quando l'utente inizia a scrivere.

Elenchi puntati in HTML

Gli elenchi puntati sono un modo semplice e intuitivo per organizzare delle informazioni in una lista. In HTML, utilizziamo i tag <ul> e <li> per creare questi elenchi.

- **<ul>:** Questo tag indica l'inizio e la fine di un elenco non ordinato (un elenco puntato). La "u" sta per "unordered".
- **<li>:** Questo tag definisce ogni singolo elemento dell'elenco. La "li" sta per "list item".

Esempio:

HTML

```
<ul>
  <li>Mela</li>
  <li>Banana</li>
  <li>Arancia</li>
</ul>
```

Questo codice HTML produrrà un elenco puntato con tre elementi: mela, banana e arancia.

Come funziona:

1. Il tag <ul> avvia l'elenco.
2. Ogni frutto è racchiuso all'interno di un tag <li>.
3. Il browser, quando incontra un <ul>, visualizza automaticamente dei puntini (o altri marcatori) davanti a ogni elemento <li>.

Elenchi numerati

Se invece vuoi un elenco numerato, puoi utilizzare il tag <ol> al posto di <ul>. La "o" sta per "ordered".

HTML

```
<ol>
  <li>Primo punto</li>
  <li>Secondo punto</li>
  <li>Terzo punto</li>
</ol>
```

Cos'è il CSS?

I fogli di stile a cascata (CSS) sono scritti in linguaggio CSS. Questo linguaggio è utilizzato per descrivere la presentazione di un documento scritto in HTML o XML. CSS permette di controllare il layout, i colori, i font e altri aspetti visivi delle pagine web in modo efficace.

Anche se la sintassi è simile a JavaScript:

- **CSS** è usato per definire lo stile e il layout delle pagine web. Gestisce aspetti come colori, font, layout e spaziatura.
- **JavaScript** è un linguaggio di scripting che permette di aggiungere interattività e funzionalità dinamiche alle pagine web. Con JavaScript, puoi creare contenuti interattivi, controllare media, animazioni e molto altro.

Mentre CSS si occupa dell'aspetto visivo, JavaScript si occupa del comportamento e delle interazioni degli elementi sulla pagina.

CSS è un linguaggio utilizzato per descrivere l'aspetto e la formattazione di un documento scritto in HTML. Con CSS, puoi controllare il layout, i colori, i font e molti altri aspetti visivi delle tue pagine web.

Struttura di Base di un File CSS

Un file CSS è composto da regole. Ogni regola ha un selettore e una dichiarazione. La dichiarazione è composta da una proprietà e un valore.

Esempio:

```
/* Questo è un commento in CSS */
selettore {
    proprietà: valore;
}
```

Esempio Pratico

Supponiamo di voler cambiare il colore del testo di tutti i paragrafi (<p>) in rosso. Ecco come fare:

```
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Esempio CSS</title>
    <link rel="stylesheet" href="stile.css">
</head>
<body>
    <p>Questo è un paragrafo.</p>
</body>
</html>
```

CSS (stile.css):

```
p {
    color: red;
}
```

Concetti Chiave

1. **Selettori:** Identificano gli elementi HTML da stilizzare.
   Esempi di selettori includono selettori di tipo (p, h1), selettori di classe (.classe), e selettori di ID (#id).

2. **Proprietà e Valori:** Definiscono cosa cambiare e come cambiarlo. Ad esempio, color è una proprietà e red è un valore.
3. **Classi e ID:** Le classi (.classe) possono essere utilizzate su più elementi, mentre gli ID (#id) devono essere unici per ogni elemento.
4. **Box Model:** Ogni elemento HTML è considerato come una scatola composta da margini, bordi, padding e contenuto.

Cos'è un `<div>`?

- **Definizione**: Il `<div>` è un elemento di contenitore generico usato per raggruppare altri elementi HTML. È spesso utilizzato per applicare stili CSS o per organizzare il layout della pagina.
- **Semantica**: Non ha un significato semantico intrinseco, cioè non comunica nulla sulla natura del contenuto che racchiude. Viene utilizzato principalmente per scopi di styling e layout.
- **Utilizzo**: Può contenere qualsiasi tipo di contenuto HTML, inclusi testi, immagini, altri elementi e persino altri `<div>`.

Cos'è un `<p>`?

- **Definizione**: Il `<p>` è un elemento di paragrafo, utilizzato per racchiudere blocchi di testo che costituiscono un paragrafo.
- **Semantica**: Ha un significato semantico, indicando che il contenuto è un paragrafo di testo. Questo è importante per l'accessibilità e per i motori di ricerca.
- **Utilizzo**: Può contenere solo testo e inline elements (come `<span>`, `<a>`, ecc.), ma non è destinato a contenere altri blocchi di livello superiore come `<div>` o `<h1>`.

Differenze principali

Caratteristica	`<div>`	`<p>`
Tipo di blocco	Contenitore generico	Paragrafo di testo
Semantica	Nessuna semantica intrinseca	Semantica per il testo
Contenuto	Può contenere qualsiasi elemento	Contiene solo testo e inline elements
Utilizzo	Layout e raggruppamento	Testo formattato in paragrafi

Esempio di Utilizzo di Classi e ID nei CSS

```
<html lang="it">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Esempio CSS</title>
    <link rel="stylesheet" href="stile.css">
</head>
<body>
    <p class="testo-rosso">Questo paragrafo è rosso.</p>
    <p id="testo-blu">Questo paragrafo è blu.</p>
</body>
</html>
```

CSS (stile.css):

```
.testo-rosso {
    color: red;
}

#testo-blu {
    color: blue;
}
```

Selettori di Base

- **Selettore di Tipo:** Seleziona tutti gli elementi di un tipo specifico.

```
p {
    color: blue;
}
```

- **Selettore di Classe:** Seleziona tutti gli elementi con una classe specifica.

```
.classe {
    color: red;
}
```

- **Selettore di ID:** Seleziona un elemento con un ID specifico.

```
#id {
    color: green;
}
```

Selettori di Attributo

- **Selettore di Attributo:** Seleziona elementi con un attributo specifico.

```
[type="text"] {
    border: 1px solid black;
}
```

Selettori Combinatori

- **Selettore Discendente:** Seleziona elementi discendenti di un elemento specifico.

```
div p {
    color: purple;
}
```

- **Selettore Figlio:** Seleziona elementi figli diretti di un elemento specifico.

```
div > p {
    color: orange;
}
```

- **Selettore Fratello Generale:** Seleziona tutti gli elementi fratelli di un elemento specifico

```
h1 ~ p {
    color: pink;
}
```

- **Selettore Fratello Adiacente:** Seleziona l'elemento fratello immediatamente successivo di un elemento specifico.

```
h1 + p {
    color: brown;
}
```

Selettori Pseudo-classi

- :hover**:** Seleziona un elemento quando l'utente ci passa sopra con il mouse.

```
a:hover {
    color: red;
}
```

- :first-child**:** Seleziona il primo figlio di un elemento.

```
p:first-child {
    font-weight: bold;
}
```

- :last-child**:** Seleziona l'ultimo figlio di un elemento.

```
p:last-child {
    font-style: italic;
}
```

- :nth-child(n)**:** Seleziona il figlio n-esimo di un elemento.

```
p:nth-child(2) {
    color: blue;
}
```

Selettori Pseudo-elementi

- ::before**:** Inserisce contenuto prima del contenuto di un elemento.

```
p::before {
    content: "Prima: ";
    color: gray;
}
```

- ::after**:** Inserisce contenuto dopo il contenuto di un elemento.

```
p::after {
    content: " Dopo";
    color: gray;
}
```

Alcune delle proprietà CSS più comuni e utili:

Proprietà di Testo

- color: Imposta il colore del testo.
- font-family: Specifica il font del testo.
- font-size: Imposta la dimensione del testo.
- font-weight: Definisce lo spessore del testo (es. bold).
- text-align: Allinea il testo (es. left, center, right).
- text-decoration: Aggiunge decorazioni al testo (es. underline).

Proprietà di Sfondo

- background-color: Imposta il colore di sfondo.
- background-image: Imposta un'immagine di sfondo.
- background-repeat: Definisce se e come ripetere l'immagine di sfondo.
- background-position: Imposta la posizione dell'immagine di sfondo.

Proprietà di Box Model

- margin: Imposta il margine esterno di un elemento.
- padding: Imposta il margine interno di un elemento.
- border: Definisce il bordo di un elemento.
- width: Imposta la larghezza di un elemento.
- height: Imposta l'altezza di un elemento.

Proprietà di Layout

- display: Definisce il tipo di display di un elemento (es. block, inline, flex).
- position: Imposta il metodo di posizionamento di un elemento (es. static, relative, absolute, fixed).
- top, right, bottom, left: Definiscono la posizione di un elemento posizionato.
- float: Permette di far fluttuare un elemento a sinistra o a destra.

Proprietà di Colore e Sfondo

- opacity: Imposta l'opacità di un elemento.
- background-clip: Specifica l'area di pittura dello sfondo (es. border-box, padding-box, content-box).

Proprietà di Bordo

- border-radius: Imposta gli angoli arrotondati di un bordo.
- border-width: Definisce la larghezza del bordo.
- border-style: Specifica lo stile del bordo (es. solid, dashed, dotted).
- border-color: Imposta il colore del bordo.

Proprietà di Transizione e Animazione

- transition: Definisce le transizioni tra gli stati di un elemento.
- animation: Specifica le animazioni per un elemento.

I Flexbox

Ogni proprietà ha una serie di valori che possono essere utilizzati per personalizzare l'aspetto degli elementi HTML.

Il **flexbox** (o **Flexible Box Layout**) è un modello di layout progettato per facilitare la distribuzione dello spazio tra gli elementi in un contenitore. Consente di creare layout più complessi e responsivi senza dover utilizzare float o posizionamenti complicati.

- flex-direction: Definisce la direzione degli elementi flessibili (es. row, column).
- justify-content: Allinea gli elementi lungo l'asse principale (es. flex-start, center, space-between).
- align-items: Allinea gli elementi lungo l'asse trasversale (es. flex-start, center, stretch).

Caratteristiche principali del flexbox:

1. **Orientamento**: Puoi disporre gli elementi in orizzontale o verticale.
2. **Allineamento**: Gli elementi possono essere allineati facilmente sia lungo l'asse principale che lungo l'asse trasversale.
3. **Dimensionamento**: Gli elementi possono crescere o restringersi in base allo spazio disponibile, rendendo il layout più flessibile.
4. **Ordine**: Puoi cambiare l'ordine degli elementi senza modificarne l'HTML.

Come si usa:

Per utilizzare il flexbox, devi impostare il contenitore come un contenitore flessibile usando la proprietà display: flex;. Ecco un esempio di base:

```
<style>
.container {
    display: flex; /* Imposta il contenitore come flex */
    flex-direction: row; /* Orientamento orizzontale (default) */
    justify-content: space-between; /* Spaziatura tra gli elementi */
    align-items: center; /* Allinea gli elementi verticalmente */
}


.item {
    flex: 1; /* Gli elementi crescono in base allo spazio disponibile */
}
</style>


<div class="container">
    <div class="item">Elemento 1</div>
    <div class="item">Elemento 2</div>
    <div class="item">Elemento 3</div>
</div>
```

Proprietà di Testo

- color: Specifica il colore del testo.

color: red; /* Colore rosso */

color: #00ff00; /* Colore verde in formato esadecimale */

color: rgb(0, 0, 255); /* Colore blu in formato RGB */

- font-size: Imposta la dimensione del testo.

font-size: 16px; /* Dimensione in pixel */

font-size: 1.5em; /* Dimensione relativa al font del genitore */

font-size: 120%; /* Dimensione in percentuale */

- text-align: Allinea il testo.

text-align: left; /* Allinea a sinistra */

text-align: center; /* Allinea al centro */

text-align: right; /* Allinea a destra */

Proprietà di Sfondo

- background-color: Imposta il colore di sfondo.

background-color: yellow; /* Colore giallo */

background-color: #ff00ff; /* Colore magenta in formato esadecimale */

background-color: rgba(255, 0, 0, 0.5); /* Colore rosso con opacità 50% */

- background-image: Imposta un'immagine di sfondo.

background-image: url('immagine.jpg'); /* URL dell'immagine */

background-image: linear-gradient(to right, red, yellow); /* Gradiente lineare */

Proprietà di Box Model

- margin: Imposta il margine esterno di un elemento.

margin: 10px; /* Margine di 10 pixel su tutti i lati */

margin: 10px 20px; /* 10 pixel sopra e sotto, 20 pixel a destra e sinistra */

margin: 10px 20px 30px 40px; /* Margine in senso orario: sopra, destra, sotto, sinistra */

- padding: Imposta il margine interno di un elemento.

padding: 5px; /* Padding di 5 pixel su tutti i lati */

padding: 5px 10px; /* 5 pixel sopra e sotto, 10 pixel a destra e sinistra */

padding: 5px 10px 15px 20px; /* Padding in senso orario: sopra, destra, sotto, sinistra */

- border: Definisce il bordo di un elemento.

border: 1px solid black; /* Bordo solido nero di 1 pixel */

border: 2px dashed blue; /* Bordo tratteggiato blu di 2 pixel */

border: 3px dotted green; /* Bordo puntinato verde di 3 pixel */

Proprietà di Layout

- display: Definisce il tipo di display di un elemento.

display: block; /* Elemento a blocco */

display: inline; /* Elemento in linea */

display: flex; /* Elemento flessibile */

- position: Imposta il metodo di posizionamento di un elemento.

position: static; /* Posizionamento statico (predefinito) */

position: relative; /* Posizionamento relativo */

position: absolute; /* Posizionamento assoluto */

position: fixed; /* Posizionamento fisso */

Esercizio:

creare pagine in HTML e CSS

CSS vs Bootstrap

Quando Usare Uno o l'Altro:

- **Solo CSS**: Ideale se hai tempo e vuoi un design completamente personalizzato. Utile per progetti di apprendimento e sperimentazione con stili avanzati.
- **Bootstrap**: Perfetto per prototipi rapidi, progetti con scadenze strette e siti che necessitano di un design pulito e responsive con il minimo sforzo.

Utilizzare solo CSS rispetto all'usare Bootstrap per creare un sito web comporta diverse differenze, ognuna con i propri vantaggi e svantaggi. Ecco una panoramica:

CSS

Vantaggi:

- **Flessibilità Totale**: Hai il controllo completo del design e dello stile del tuo sito. Puoi personalizzare ogni aspetto senza restrizioni imposte da un framework.

- **Leggerezza**: Il codice CSS personalizzato può essere più leggero e veloce da caricare, poiché non include funzionalità non necessarie.
- **Apprendimento Profondo**: Lavorare direttamente con CSS ti permette di comprendere meglio come funzionano i layout, il posizionamento e gli stili.

Svantaggi:

- **Tempo**: Creare un design completamente da zero richiede più tempo e può essere complesso.
- **Compatibilità**: Potrebbe essere necessario scrivere codice aggiuntivo per garantire la compatibilità cross-browser.

Bootstrap

Vantaggi:

- **Semplicità e Velocità**: Bootstrap è un framework che fornisce componenti pronti all'uso e stili predefiniti, permettendoti di creare un sito attraente rapidamente.
- **Responsive**: Include già un sistema di griglie responsive, facilitando la creazione di layout adattabili a diversi dispositivi.
- **Comunità e Supporto**: Essendo ampiamente utilizzato, ci sono molte risorse, documentazione e supporto disponibile.

Svantaggi:

- **Rigidità**: Può essere limitante se desideri un design altamente personalizzato. Potresti dover sovrascrivere molti stili predefiniti.
- **Peso**: Caricare il framework completo può aggiungere peso al tuo sito, influenzando i tempi di caricamento.

JAVASCRIPT

JavaScript è un linguaggio di scripting che ti permette di aggiungere interattività e funzionalità dinamiche ai siti web. Ecco alcune risorse e concetti chiave per iniziare a capire JavaScript:

- Introduzione a JavaScript
  - Sintassi di base
  - Variabili (let, const)
  - Tipi di dati (stringhe, numeri, booleani)
- Operatori e Condizionali
  - Operatori aritmetici e logici
  - If, else, else if
- Cicli
  - For, while, do...while

Funzioni e Oggetti

- Funzioni
  - Dichiarazione e invocazione di funzioni
  - Parametri e valori di ritorno
- Array
  - Creazione e manipolazione di array
  - Metodi utili degli array (push, pop, shift, unshift, map, filter)
- Oggetti
  - Creazione di oggetti
  - Prototipi e ereditarietà

Manipolazione del DOM

- Selezione e Modifica di Elementi DOM
  - getElementById, querySelector
  - Modifica di testo e attributi
- Eventi
  - Gestione di eventi (click, mouseover, keypress)
  - Event listeners (addEventListener)
- Esempi Pratici
  - Creazione di un semplice progetto che utilizza DOM e Eventi

Concetti Chiave

1. **Variabili**: Usate per memorizzare dati che possono essere utilizzati e manipolati nel tuo script.

```
let nome = "Mario";
const età = 25;
```

2. **Funzioni**: Blocchi di codice che possono essere riutilizzati e eseguiti quando necessario.

```
function saluta() {
   console.log("Ciao, mondo!");
}
saluta(); // Chiamata alla funzione
```

3. **Eventi**: Azioni che accadono nel browser (come un click di un pulsante) alle quali puoi rispondere con JavaScript.

```
document.getElementById("mioBottone").addEventListener("click", function() {
   alert("Bottone cliccato!");
});
```

4. **Manipolazione del DOM**: Modifica dinamica del contenuto HTML e degli stili.

```
document.getElementById("mioParagrafo").innerHTML = "Testo aggiornato!"
```

Impara le Basi:

- **Variabili**: Come memorizzare e manipolare i dati.
- **Operatori**: Come eseguire operazioni matematiche e logiche.
- **Condizionali**: if e else per prendere decisioni nel codice.
- **Cicli**: for, while per eseguire il codice ripetutamente.
- **Funzioni**: Blocchi di codice riutilizzabili.

La sintassi di base di JavaScript

è piuttosto semplice e simile a molti altri linguaggi di programmazione:

1. Dichiarazione di Variabili

```
let nome = "Mario";  // Può essere modificato
const eta = 30;      // Costante, non può essere modificata
var citta = "Roma";  // Non raccomandato per l'uso moderno, preferisci `let` o `const`
```

2. Operatori Aritmetici e Logici

```
let somma = 5 + 3;
let prodotto = 5 * 3;
let confronto = 5 > 3; // true
```

3. Strutture Condizionali

```
if (eta > 18) {
    console.log("Sei un adulto.");
} else {
    console.log("Sei un minorenne.");
}
```

4. Cicli

```
for (let i = 0; i < 5; i++) {
    console.log(i);  // Stampa i numeri da 0 a 4
}

let j = 0;
while (j < 5) {
    console.log(j);  // Stampa i numeri da 0 a 4
    j++;
}
```

5. Funzioni

```
function saluta(nome) {
    return "Ciao, " + nome + "!";
}

console.log(saluta("Mario"));  // Stampa "Ciao, Mario!"
```

6. Oggetti

```
let persona = {
    nome: "Mario",
```

```
    eta: 30,
    saluta: function() {
        return "Ciao, " + this.nome + "!";
    }
};

console.log(persona.saluta());  // Stampa "Ciao, Mario!"
```

7. Array

```
let numeri = [1, 2, 3, 4, 5];
console.log(numeri[0]);  // Stampa 1

// Aggiungi un elemento all'array
numeri.push(6);
console.log(numeri);  // Stampa [1, 2, 3, 4, 5, 6]
```

8. Manipolazione del DOM (Il DOM, acronimo di Document Object Model, è una rappresentazione strutturata di un documento HTML**)**

```
document.getElementById("mioID").innerHTML = "Nuovo testo!";
```

9. Gestione degli Eventi

```
document.getElementById("mioBottone").addEventListener("click", function() {
    alert("Bottone cliccato!");
});
```

JavaScript supporta un'ampia gamma di eventi che possono essere intercettati per creare pagine web.

Alcuni degli attributi di eventi HTML più comuni che puoi utilizzare direttamente nel markup HTML per rendere la tua pagina interattiva:

Eventi di Mouse

- onclick: Eseguito quando l'elemento viene cliccato.

```
<button onclick="myFunction()">Clicca qui</button>
```

- ondblclick: Eseguito quando l'elemento viene doppio cliccato.

```
<button ondblclick="myFunction()">Doppio clicca qui</button>
```

- onmousedown: Eseguito quando il pulsante del mouse viene premuto sull'elemento.

```
<div onmousedown="myFunction()">Mouse giù</div>
```

- onmouseup: Eseguito quando il pulsante del mouse viene rilasciato sull'elemento.

```
<div onmouseup="myFunction()">Mouse su</div>
```

- onmouseover: Eseguito quando il puntatore del mouse passa sopra l'elemento.

```
<div onmouseover="myFunction()">Mouse sopra</div>
```

- onmouseout: Eseguito quando il puntatore del mouse esce dall'elemento.

```
<div onmouseout="myFunction()">Mouse fuori</div>
```

Eventi di Tastiera

- onkeydown: Eseguito quando un tasto viene premuto.

```
<input type="text" onkeydown="myFunction()">
```

- onkeypress: Eseguito quando un tasto viene premuto e rilasciato.

```
<input type="text" onkeypress="myFunction()">
```

- onkeyup: Eseguito quando un tasto viene rilasciato.

```
<input type="text" onkeyup="myFunction()">
```

Eventi di Finestra

- onload: Eseguito quando la pagina è completamente caricata.

```
<body onload="myFunction()">
```

- onresize: Eseguito quando la finestra del browser viene ridimensionata.

```
<body onresize="myFunction()">
```

- onscroll: Eseguito quando la pagina viene scorsa.

```
<body onscroll="myFunction()">
```

Eventi di Form

- onsubmit: Eseguito quando un modulo viene inviato.

```
<form onsubmit="myFunction()">
```

- onreset: Eseguito quando un modulo viene resettato.

<form onreset="myFunction()">

- onchange: Eseguito quando il valore di un elemento di input cambia.

<input type="text" onchange="myFunction()">

- onfocus: Eseguito quando un elemento ottiene il focus.

<input type="text" onfocus="myFunction()">

- onblur: Eseguito quando un elemento perde il focus.

<input type="text" onblur="myFunction()">

Eventi Multimediali

- onplay: Eseguito quando la riproduzione di un video o audio inizia.

<video onplay="myFunction()">

- onpause: Eseguito quando la riproduzione di un video o audio è messa in pausa.

<video onpause="myFunction()">