Programmazione lato client

Prof. Luca Campion

November 9, 2025

Contents

Chapter 1

Introduzione a HTML5

1.1 Cos'è HTML?

HTML (HyperText Markup Language) è il linguaggio di marcatura per creare pagine web. Non è un linguaggio di programmazione come Java (vedi il capitolo sulla programmazione orientata agli oggetti nel corso di Quarta), ma un modo per strutturare e descrivere il contenuto usando tag e attributi.

1.2 Struttura base di un documento HTML5

1.2.1 Anatomia dei tag HTML

Un tag HTML è composto da:

- **Tag di apertura**: `<nome>`
- **Contenuto**: testo o altri tag annidati
- **Tag di chiusura**: `</nome>`

Esempio di tag con contenuto:

```
<p>Questo è un paragrafo</p>
<div>
    <p>Questo paragrafo è annidato nel div</p>
</div>
```

1.2.2 Tag auto-chiudenti (self-closing tags)

Alcuni elementi HTML non hanno contenuto. Questi sono chiamati **tag auto-chiudenti** o **self-closing tags**. Per garantire la massima compatibilità con XHTML e XML, è importante chiudere esplicitamente questi tag:

```
<img src="immagine.jpg" alt="Una descrizione" />
<input type="text" placeholder="Inserisci testo" />
<meta charset="UTF-8" />
```

```
<link rel="stylesheet" href="stile.css" />
<area shape="rect" coords="0,0,100,100" href="link.html" />
```

Nota

Anche se HTML5 permette di omettere la chiusura (`/>`), è consigliabile usare sempre la sintassi XML/XHTML con chiusura esplicita per:

- Garantire la compatibilità con XHTML e XML
- Rendere il codice più consistente e leggibile
- Evitare problemi con parser XML
- Facilitare la migrazione tra diversi formati

Nota

Nota: In passato si utilizzavano i tag `<br>` e `<hr>` per creare interruzioni di riga e linee orizzontali. Queste pratiche sono ora considerate obsolete poiché mescolano presentazione e contenuto. È preferibile utilizzare CSS per gestire lo spazio e le separazioni visive.
Invece di `<br>`, usa:

- Elementi blocco appropriati (`<p>`, `<div>`)
- CSS `margin` o `padding`
- Proprietà CSS `white-space`

Invece di `<hr>`, usa:

- CSS `border`
- Elementi semantici con stili appropriati
- `<section>` o altri elementi strutturali

1.2.3 Commenti HTML

I commenti in HTML sono utili per documentare il codice e vengono ignorati dal browser. La sintassi è:

```
<!-- Questo è un commento -->
<!--
    I commenti possono
    essere supiùrighe
-->
```

I commenti sono utili per:

- Documentare sezioni complesse del codice
- Disabilitare temporaneamente parti di HTML
- Aggiungere note per altri sviluppatori

1.2.4 Attributi HTML

Gli attributi forniscono informazioni aggiuntive ai tag e hanno questa sintassi:

```
<tag attributo="valore">contenuto</tag>
```

1.2.5 Gli attributi id e class

Gli attributi `id` e `class` sono fondamentali per identificare e raggruppare elementi HTML:

`id` È un identificatore **univoco** per un elemento nella pagina:

- Deve essere unico nel documento (non ci possono essere due elementi con lo stesso `id`)
- Viene usato per identificare un elemento specifico
- Utile per JavaScript e per collegamenti interni alla pagina
- Si riferisce con `#` nel CSS (es: `#header`)

Esempio: `<div id="header">`

`class` Definisce una o più categorie a cui l'elemento appartiene:

- Un elemento può avere multiple classi (separate da spazi)
- La stessa classe può essere usata su più elementi
- Utile per applicare stili comuni a gruppi di elementi
- Si riferisce con `.` nel CSS (es: `.container`)

Esempio: `<div class="container blue">`

Nota

Differenze principali:

- `id`: come la carta d'identità, è unico e identifica un singolo elemento
- `class`: come un gruppo o categoria, può essere condiviso tra più elementi

1.2.6 Attributi per collegamenti e risorse

L'attributo href

L'attributo `href` (Hypertext REFerence) definisce il collegamento a una risorsa:

- Usato principalmente con il tag `<a>` per creare link
- Può contenere:
  - URL assoluti: `https://www.esempio.com`
  - URL relativi: `./immagini/foto.jpg`

  - Collegamenti interni: `#sezione`
  - Collegamenti email: `mailto:esempio@email.com`
  - Collegamenti telefonici: `tel:+390123456789`

Esempi di utilizzo:

```
<!-- Link a sito web -->
<a href="https://www.esempio.com/">Visita il sito</a>

<!-- Link a sezione della pagina -->
<a href="#introduzione">Vai all'introduzione</a>

<!-- Link a email -->
<a href="mailto:info@esempio.com">Contattaci</a>
```

L'attributo src

L'attributo `src` (source) specifica la fonte di una risorsa multimediale:

- Utilizzato con elementi che caricano contenuti esterni:
  - `<img>` per immagini
  - `<video>` per filmati
  - `<audio>` per suoni
  - `<script>` per JavaScript
- Può contenere:
  - Percorsi assoluti: `https://cdn.esempio.com/immagine.jpg`
  - Percorsi relativi: `./assets/logo.png`
  - Data URI: `...`

Esempi di utilizzo:

```
<!-- Immagine da URL assoluto -->
<img src="https://esempio.com/foto.jpg" alt="Descrizione" />

<!-- Immagine da percorso relativo -->
<img src="./immagini/logo.png" alt="Logo" />

<!-- Video con sorgenti multiple -->
<video controls="controls">
    <source src="video.mp4" type="video/mp4" />
    <source src="video.webm" type="video/webm" />
</video>
```

Nota

Differenze principali:

- `href`: per collegamenti e riferimenti (dove vuoi andare)
- `src`: per contenuti da incorporare (cosa vuoi mostrare)

1.2.7 Annidamento dei tag

I tag possono essere annidati (contenuti dentro altri tag) ma devono seguire regole precise:

- I tag devono essere chiusi nell'ordine inverso di apertura
- Ogni tag figlio deve essere completamente contenuto nel genitore
- L'indentazione aiuta a visualizzare la struttura

Esempio corretto:

```
<div class="container">
    <header>
        <h1>Titolo</h1>
        <nav>
            <a href="pagina1.html">Link 1</a>
            <a href="/about/index.html">Chi siamo</a>
            <a href="../contatti.html">Contatti</a>
        </nav>
    </header>
</div>
```

Esempio errato:

```
<div><p>Questo è <!-- NON FARE COSÌ -->
    <span>sbagliato</div></p></span>
```

Attenzione

L'annidamento errato può causare comportamenti imprevedibili nel rendering della pagina e non passa la validazione HTML.

1.2.8 Commenti HTML

I commenti in HTML servono per documentare il codice e non vengono visualizzati nel browser. Sono utili per:

- Spiegare sezioni complesse di codice
- Disabilitare temporaneamente parti di codice
- Organizzare il codice in sezioni logiche

- Comunicare con altri sviluppatori

Sintassi dei commenti:

```
<!-- Questo è un commento su una riga -->

<!--
  Questo è un commento
  su più righe
-->

<!-- Non usare -- dentro i commenti -->

<div class="header">
  <!-- TODO: aggiungere logo -->
  <h1>Titolo</h1>
</div>

<!--[if IE 8]>
  <link href="ie8only.css" rel="stylesheet">
<![endif]-->
```

Nota

I commenti HTML possono contenere qualsiasi testo tranne - doppio trattino, che può causare problemi di parsing.

Attenzione

I commenti sono visibili nel codice sorgente della pagina. Non inserire informazioni sensibili (password, chiavi API, ecc.) nei commenti.

1.2.9 Struttura completa documento

Ecco la struttura completa di un documento HTML5:

```
<!DOCTYPE html>
<html lang="it">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale
      =1.0" />
  <title>Titolo della pagina</title>
  <link rel="stylesheet" href="style.css" />
</head>
<body>
  <header>
    <h1>Intestazione principale</h1>
  </header>
  <main>
    <p>Contenuto principale della pagina</p>
  </main>
  <footer>
    <p>&copy; 2025 - Tutti i diritti riservati</p>
```

```
  </footer>
</body>
</html>
```

1.2.10 Elementi principali

<!DOCTYPE html> Dichiara che il documento è HTML5, quindi deve rispettare le regole di questa versione del linguaggio.

<html> Elemento radice del documento

<head> Contiene metadati e link a risorse

<body> Contiene il contenuto visibile

1.3 Tag semantici

HTML5 introduce tag semantici che descrivono il significato del contenuto:

```
<header>Intestazione del sito</header>
<nav>Barra di navigazione</nav>
<main>Contenuto principale</main>
<section>Una sezione tematica</section>
<article>Un articolo indipendente</article>
<aside>Barra laterale</aside>
<footer>Piè di pagina</footer>
```

Attenzione

La semantica HTML è importante per l'accessibilità: i lettori di schermo leggono il significato dei tag, non solo il testo.

1.4 Meta tag essenziali

I meta tag sono elementi HTML che forniscono metadati sul documento HTML. Questi tag non sono visualizzati nella pagina ma contengono informazioni importanti per browser, motori di ricerca e altri servizi web.

1.4.1 Sintassi generale

Un meta tag ha questa struttura:

```
<meta name="nome" content="valore">
<!-- oppure -->
<meta http-equiv="nome" content="valore">
<!-- oppure -->
<meta charset="codifica">
```

1.4.2 Meta tag fondamentali

Ecco i meta tag più importanti e il loro utilizzo:

```
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale
    =1.0" />
<meta name="description" content="Breve descrizione della pagina"
    />
<meta name="keywords" content="html, css, web, sviluppo" />
```

charset Specifica la codifica dei caratteri del documento. UTF-8 è lo standard moderno che supporta caratteri internazionali e emoji.

viewport Controlla come la pagina si adatta ai dispositivi mobili:

- `width=device-width`: imposta la larghezza della pagina alla larghezza del dispositivo
- `initial-scale=1.0`: imposta il livello di zoom iniziale

description Fornisce una breve descrizione della pagina (massimo 155 caratteri). Viene mostrata nei risultati di ricerca di Google.

keywords Elenca parole chiave rilevanti per la pagina. Oggi ha meno importanza per il SEO ma è ancora utilizzato da alcuni motori di ricerca.

Attenzione

Il meta tag viewport è cruciale per il responsive design. Senza di esso, i siti mobile-friendly non funzioneranno correttamente sui dispositivi mobili.

Nota

Il meta tag `viewport` è fondamentale per il responsive design. Senza di esso, i dispositivi mobili non visualizzeranno la pagina correttamente.

1.5 Riepilogo

- HTML è il linguaggio di marcatura del web
- `<!DOCTYPE html>` identifica la versione HTML5
- Tag semantici descrivono il significato del contenuto
- Meta tag sono fondamentali per accessibilità e responsive
- Sempre validare il codice HTML

Chapter 2

Tag di Blocco e Riga

2.1 Differenza tra blocco e riga

In HTML, i tag sono classificati in due categorie:

Block-level L'elemento occupa tutta la larghezza disponibile e inizia su una nuova riga

Inline L'elemento occupa solo lo spazio necessario e rimane sulla stessa riga

2.2 Tag di blocco

I tag di blocco iniziano su una nuova riga:

```
<!-- Esempio di elementi di blocco HTML -->
<div class="container">Contenitore generico</div>
<p>Questo è un paragrafo di testo</p>
<!-- I titoli hanno diversi livelli di importanza -->
<h1>Titolo principale</h1>
<h2>Titolo secondario</h2>
<section>Una sezione tematica</section>
<article>Un articolo completo</article>
<!-- Le liste possono essere ordinate o non ordinate -->
<ul>
  <li>Lista elemento non ordinato</li>
  <li>Lista elemento non ordinato</li>
</ul>
<ol>
  <li>Lista elemento ordinato</li>
  <li>Lista elemento ordinato</li>
</ol>
```

2.2.1 Tag blocco comuni

- `<div>`: Contenitore generico
- `<p>`: Paragrafo

- `<h1>` a `<h6>`: Heading (titoli)
- `<section>`, `<article>`, `<header>`, `<footer>`: Semantici
- `<ul>`, `<ol>`, `<li>`: Liste
- `<blockquote>`: Citazione in blocco

2.3 Tag inline

I tag inline non creano nuove righe e occupano solo lo spazio del contenuto:

```
<span class="highlight">testo importante</span>
<a href="https://example.com">collegamento</a>
<strong>testo forte (bold)</strong>
<em>testo enfatizzato (italic)</em>
<code>codice inline</code>
<img src="immagine.jpg" alt="Descrizione" />
```

2.4 Proprietà display CSS

La proprietà CSS `display` controlla come un elemento viene visualizzato:

```
/* Block-level */
display: block;
display: flex;
display: grid;

/* Inline */
display: inline;
display: inline-block;
display: none;
```

Nota

Con `display: inline-block`, un elemento inline può avere larghezza e altezza come un blocco, ma rimane sulla stessa riga degli elementi vicini.

2.5 Riepilogo

- Tag blocco occupano tutta la larghezza disponibile
- Tag inline occupano solo lo spazio del contenuto
- `display: block/inline/inline-block` controlla il comportamento
- Proprietà CSS può modificare il comportamento di default
- Semantica HTML rimane importante indipendentemente da display

Chapter 3

Form e Input

3.1 Introduzione ai Form

I form (moduli) sono elementi fondamentali del web che permettono l'interazione tra utente e sito web. Sono utilizzati per:

- Raccogliere dati dagli utenti (registrazione, login, contatti)
- Effettuare ricerche
- Caricare file
- Inviare feedback o commenti
- Completare transazioni (acquisti, prenotazioni)
- Configurare preferenze

Il processo di gestione di un form prevede tre fasi:

1. **Raccolta dati**: L'utente compila i campi del form
2. **Invio**: I dati vengono inviati al server tramite una richiesta HTTP
3. **Elaborazione**: Il server processa i dati e restituisce una risposta

Nota

I form sono il ponte tra il frontend (interfaccia utente) e il backend (logica del server). La sicurezza e la validazione dei dati sono cruciali in questo processo.

3.2 Elemento form

Il tag `<form>` crea un modulo per raccogliere dati dall'utente. È simile ai metodi di una classe Java (vedi il corso di Quarta sulla programmazione orientata agli oggetti) che accettano parametri:

```
<form action="paginaDiArrivo.html" method="POST" name="contactForm"
    >
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required>

  <button type="submit">Invia</button>
  <button type="reset">Cancella</button>
</form>
```

3.2.1 Attributi form

action URL dove inviare i dati del form

method GET o POST (modalità invio dati)

name Nome identificativo del form

enctype Tipo di codifica (application/x-www-form-urlencoded, multipart/form-data)

3.3 Input types

HTML5 offre molti tipi di input per diversi dati:

```
<input type="text" placeholder="Testo generico" />
<input type="password" placeholder="Password" />
<input type="email" placeholder="Email" />
<input type="number" min="0" max="100" step="5" />
<input type="date" />
<input type="checkbox" id="terms" /> <label for="terms">Accetto i
    termini</label>
<input type="radio" name="option" value="1" id="opt1" /> <label for
    ="opt1">Opzione 1</label>
<select>
  <option>Scegli un'opzione</option>
</select>
<textarea rows="5" cols="40"></textarea>
```

3.4 Label e accessibilità

Il tag `<label>` associa il testo all'input:

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">

<label for="subscribe">Iscrivimi alla newsletter</label>
  <input type="checkbox" id="subscribe" name="subscribe">
```

Nota

L'attributo `for` del label deve corrispondere all'attributo `id` dell'input per accessibilità screen reader.

3.5 Validazione HTML5

HTML5 supporta validazione lato client:

```
<input type="email" required />
<input type="number" min="0" max="10" />
<input type="text" pattern="[A-Za-z]+" title="Solo lettere" />
<input type="password" minlength="8" />
```

Attenzione

La validazione HTML5 non sostituisce la validazione server-side. Sempre convalidare i dati sul server poiché l'utente potrebbe modificare il codice client.

3.6 Riepilogo

- I form sono lo strumento principale per l'interazione utente-server
- L'elemento `<form>` ha attributi essenziali:
  - `action`: dove inviare i dati
  - `method`: come inviarli (GET/POST)
  - `enctype`: tipo di codifica
- HTML5 offre diversi tipi di input specializzati:
  - `text`, `email`, `password`, `number`
  - `date`, `checkbox`, `radio`
  - `file`, `textarea`, `select`
- Ogni `input` dovrebbe avere un `label` associato
- La validazione client-side (HTML5) va sempre accompagnata da validazione server-side
- L'accessibilità è fondamentale (label, attributi `for` e `id`)

Chapter 4

Bibliografia e Risorse

4.1 Documentazione ufficiale

Consulta MDN Web Docs per HTML e CSS, W3C per specifiche tecniche, Can I Use per compatibilità browser.

4.2 Tutorial e Learning

CodePen, JsFiddle, CSS-Tricks per articoli e tutorial. Flexbox Froggy e Grid Garden per giochi interattivi.

4.3 Tool online

W3C HTML Validator, W3C CSS Validator, WebAIM WCAG Checker per validazione. Sass Playground per compilazione SCSS online.

4.4 Editor e IDE

VS Code (consigliato con Live Server), Sublime Text, WebStorm, Atom per sviluppo HTML/CSS.

4.5 Editor e IDE

- **VS Code** (Gratuito) - Raccomandato
  - Estensioni: Live Server, CSS Intellisense, Prettier
- **Sublime Text** (Pagato, prova gratuita)
- **WebStorm** (Pagato, ma completo)
- **Atom** (Gratuito, da GitHub)

4.6 Browser Developer Tools

- **Chrome DevTools**: F12 o Ctrl+Shift+I
- **Firefox Developer Edition**: F12
- **Safari Web Inspector**: Cmd+Option+I
- Strumenti essenziali: Inspector, Console, Network, Performance

4.7 Libri consigliati

- "HTML & CSS: Design and Build Websites" - Jon Duckett
- "Responsive Web Design" - Ethan Marcotte
- "CSS Secrets" - Lea Verou
- "The Pragmatic Programmer" - (contiene best practices web)

4.8 Community e Forum

- **Stack Overflow**: `https://stackoverflow.com/` - Q&A
- **CSS-Tricks Community**: Forum e discussioni
- **Reddit r/webdev**: Comunità web developers
- **GitHub**: Cerca progetti open source per imparare

4.9 Collegamento con altri corsi

Come visto in questo corso:

- **Terza (Linguaggio C)**: Introduzione a C - Logica e algoritmi
- **Quarta (Java)**: Classi, Oggetti, Ereditarietà e Package - OOP e backend
- **Quarta (HTML/CSS)**: Questo corso - Frontend web

—

Ultimo aggiornamento: Novembre 2025