

1 Lezione del 23-09-24

1.1 Introduzione

Il corso di progettazione web tratta i seguenti argomenti:

- Il linguaggio HTML 5.0
- I Cascading Style Sheet (CSS)
- Programmazione lato client: JavaScript (ECMAScript)
- Programmazione lato server: PHP
- Il protocollo HTTP

1.2 Storia del Web

Il Web, o World-Wide Web, è un'applicazione di Internet (come lo sono la posta elettronica, il File Transfer Protocol (FTP), ecc...).

1.2.1 Reti a commutazione

Le reti telefoniche (vecchio stile) forniscono un buon modello per il funzionamento di Internet. Le vecchie reti telefoniche erano reti a **commutazione di circuito**, ergo stabilivano una connessione fra due utenti attraverso un circuito fisico. Il filo di rame fisico che collegava i due utenti era riservato a loro, e non poteva servire nessun'altro durante il tempo della chiamata. Questo sistema aveva delle chiare problematiche:

- La connessione va stabilita e mantenuta per la durata della chiamata;
- C'è difficoltà a mantenere più chiamate simultaneamente;
- Si spreca banda in quanto anche i silenzi vengono trasmessi.

Un'alternativa è una rete a **commutazione di pacchetto**. La rete ARPANET non usava una rete a commutazione di circuito, ma una rete a commutazione di pacchetti. In una rete di questo tipo, il messaggio viene diviso in pacchetti, che possono prendere strade diverse per raggiungere il destinatario. Un pacchetto contiene informazioni sul mittente, il destinatario, e l'ordine del pacchetto stesso nel messaggio trasmesso.

Ad esempio, se Aldo vuole inviare a Bruno il messaggio "Didgeridoo è una parola difficile da dire", questo messaggio potrebbe essere diviso nei pacchetti:

- Aldo | Bruno | 1 | "Didgeridoo"
- Aldo | Bruno | 2 | "è una parola"
- Aldo | Bruno | 3 | "difficile da dire"

e questi pacchetti tramessi attraverso la rete dal mittente al destinatario.

Le reti a commutazioni di pacchetti sono più efficienti in quanto:

- Sono più robuste;
- Non richiedono la totalità delle risorse di trasferimento finché si mantiene la conversazione.

Internet nasce dal progetto ARPANET, e adotta questo tipo di design.

1.2.2 Protocollo TCP/IP

Uno dei primi problemi sorti nella progettazione di Internet è stato quello di unificare le regole di dialogo fra più reti. Una prima soluzione a questo problema, sorta nel 1981, è stata il Transmission Control Protocol / Internet Protocol (TCP/IP).

1.2.3 Web

L'invenzione del Web si attribuisce solitamente a Tim Berners-Lee. Dalla sua concezione iniziale ad oggi, le caratteristiche fondamentali del sistema sono rimaste invariate:

- Un Unique Resource Locator (URL) per identificare e locare risorse sul Web;
- Il protocollo HTTP per descrivere come le richieste e le risposte operano;
- Un programma, chiamato **server**, che risponde alle richieste HTTP;
- Il linguaggio HTML per pubblicare documenti;
- Un programma, chiamato **browser** (che fa la parte del **client**), che crea richieste HTTP dagli URL, e visualizza l'HTML che riceve.

Poco dopo la sua fondazione, Berners-Lee aiutò a fondare il Word Wide Web Consortium (W3C), che ha facilitato la crescita della rete attraverso la creazione di standard. Questo è stato permesso dalla decisione del CERN di rendere tutto il codice sorgente e i protocolli web sviluppati in questo periodo, effettivamente liberi.

Alcuni dei vantaggi delle applicazioni Web sono:

- Accessibilità da parte di ogni computer connesso a Internet;
- Utilizzabile con più browser e sistemi operativi;
- Più semplici da aggiornare;
- Archiviazione dei dati centralizzata.

alcuni contro, invece, sono:

- Richiesta di avere una connessione internet attiva.
- Problemi di sicurezza, dati dalla trasmissione di dati sensibili attraverso Internet;
- Problemi di inaderenza agli standard di alcuni siti su alcuni browser;
- Accesso limitato al sistema operativo.

1.2.4 Intranet

Un'intranet è una rete internet limitata ad un'organizzazione o un'azienda. A differenza delle reti internet, le reti intranet sono private e limitate agli utenti che le usano (impiegati o clienti). Il controllo sulle connessioni in entrata e in uscita di una rete internet o intranet è solitamente eseguito da un **firewall**.

1.2.5 Siti Web statici

Un sito web statico è formato da pagine HTML che appaiono identiche a tutti gli utenti in tutti i momenti.

Storicamente, questi contenuti venivano mantenuti dal cosiddetto webmaster, che doveva solo modificare il codice in HTML e, eventualmente, immagini o altri contenuti multimediali.

Il principio di funzionamento di un sito statico è il seguente:

- L'utente (il client) richiede una pagina al server;
- Il server risponde alla richiesta erogando la pagina desiderata;
- L'utente riceve la pagina e la visualizza.

1.2.6 Siti Web dinamici

Dopo alcuni anni dall'invenzione del Web, alcuni siti iniziarono a diventare più complicati e ad usare programmi che giravano sui web server per creare contenuti in maniera dinamica.

Secondo questo modello, il principio di funzionamento di un sito statico è il seguente:

- L'utente (il client) richiede una pagina al server;
- Il server risponde alla richiesta erogando la pagina desiderata e eventualmente riempiendola di informazioni generate da script, programmi, ecc... eseguiti sul momento;
- L'utente riceve la pagina e la visualizza.

1.2.7 Web 2.0

Web 2.0 è una buzzword che si riferisce principalmente a due cose:

- Per gli utenti, Web 2.0 significa applicazioni interattive dove si possono consumare ma anche creare nuovi contenuti;
- Per gli sviluppatori, Web 2.0 significa un nuovo modo di creare siti Web dinamici: la logica di programmazione si spostava dal server al client, attraverso il linguaggio Javascript. Questo introduceva diversi problemi per quanto riguarda l'esecuzione di comunicazioni asincrone.

1.3 Protocolli Internet

Un protocollo è una serie di regole che vengono usate da più calcolatori che comunicano fra loro. I protocolli TCP/IP vennero inizialmente pensati come uno stack di quattro livelli, anche se alcune astrazioni dividono questo stack ulteriormente in quattro o cinque livelli. Vediamo quindi i quattro livelli fondamentali, dal basso verso l'alto:

1. **Link Layer:** si occupa della trasmissione fisica di bit di informazione.
Esempio: **MAC**;
2. **Internet Layer:** stabilisce connessioni, routing e indirizzamento.
Esempio: **IPv4, IPv6**;
3. **Transport Layer:** si assicura che le trasmissioni arrivino in ordine e senza errori.
Esempio: **TCP, UDP**;
4. **Application Layer:** protocolli che permettono alle applicazioni di interagire con il Transport Layer.
Esempio: **HTTP, FTP, POP**.

1.3.1 Link Layer

Il link layer permette il controllo sull'hardware che stabilisce un collegamento con i mezzi di trasmissione cablata (vedi Ethernet) o wireless (vedi Wi-Fi), ovvero il Media Access Control (MAC), e fornisce gestione di errori e di flusso, ovvero il Logical Link Control (LLC).

1.3.2 Internet Layer, protocollo IP

Il protocollo IP identifica destinazioni attraverso indirizzi IP. Ogni dispositivo connesso a internet ha un indirizzo IP proprio, ovvero un codice di 32 bit che lo identifica univocamente.

In particolare, gli indirizzi IPv4 sono quelli del protocollo TCP/IP originale. La rappresentazione comune degli indirizzi IPv4 (192.168.0.1) racchiude fra punti interi di 8 bit per volta dei 32 dell'indirizzo.

Gli indirizzi IPv6 appartengono ad un protocollo aggiornato, attualmente in corso di adozione, che adottano 8 interi da 16 bit. Solitamente gli interi del protocollo IPv6 vengono scritti in esadecimale.

L'indirizzo IP viene solitamente assegnato da un Internet Service Provider (ISP). Su reti locali, più computer possono condividere un'indirizzo IP, attraverso il NAT. NAT sta per Network Address Translation. All'interno della rete locale, ogni dispositivo usa un indirizzo diverso e privato (10.0.0/24). In uscita dalla rete locale, tutti i datagrammi hanno lo stesso indirizzo NAT IP di mittente, e diversi numeri di port. Questo è permesso da un meccanismo interno al router NAT che converte automaticamente indirizzi IP e numeri di porta in indirizzi IP locali, e viceversa.

1.3.3 Transport Layer, protocollo TCP

L'IP ha diverse limitazioni:

- La consegna dei pacchetti non è garantita;

- I pacchetti potrebbero arrivare in disordine;
- La comunicazione avviene fra dispositivi;
- Nessun controllo sul flusso o sulle congestioni.

Il Transport Layer si assicura che ogni trasmissione arrivi in ordine e senza errori, attraverso dei meccanismi di sicurezza:

1. Prima i dati sono divisi in pacchetti formattati secondo il protocollo TCP;
2. In seguito, ogni pacchetto viene restituito al mittente attraverso l'ACK (acknowledge).

In questo modo, il mittente sa se i suoi dati sono stati effettivamente ricevuti, e può reinviare i pacchetti perduti.

Il protocollo TCP fornisce una serie di altre funzionalità: la gestione del flusso e della congestione. Questi meccanismi riguardano la diminuzione della frequenza di trasmissione in caso di sovraccarico del destinatario.

1.3.4 Application Layer

L'application layer è un livello di astrazione. Fanno parte di questo livello il protocollo di trasferimento di pagine web, ovvero l'Hypertext Transfer Protocol (HTTP), i protocolli di trasmissione di file, ovvero il File Transfer Protocol (FTP), o i protocolli di trasmissione di posta elettronica, come il Post Office Protocol (POP), fra gli altri.

1.4 Modello client-server

Nel modello client-server, i dispositivi si dividono in client e server. Il server è attivo e pronto ad erogare servizi, mentre il client si occupa di fare richieste a cui il server dovrà adempire. Questo meccanismo forma il cosiddetto request-response loop.

1.4.1 L'alternativa peer-to-peer

In una rete peer-to-peer non esiste un'unità centrale (il server), ma ogni nodo è in grado di inviare e ricevere indipendentemente dagli altri (peer).

Questo sistema è molto comodo per la trasmissione in massa di dati senza la possibilità di arrestare il processo a partire da una sola macchina (vedi Napster).

1.4.2 Tipi di server

Prima, abbiamo mostrato il server come una macchina singola. In verità, il server può essere formato da più macchine che si dividono il carico delle richieste dei client, e che rispondono dallo stesso URL.

Un'altra alternativa sono le cosiddette **server farm**. Una server farm distribuisce il carico delle richieste (attraverso i **load balancer**). Le server farm forniscono inoltre **fail-over redundancy**, ovvero la sicurezza che il servizio non si arresti anche nel caso di fallimento di singole macchine.

Le server farm sono tipicamente collocate all'interno di specifiche strutture dette **data center**. Non è inusuale archiviare lo stesso sito su più macchine, in parallelo. Può accadere anche il contrario: più pagine possono essere archiviate sullo stesso server.

1.5 Nozioni sull'infrastruttura

Oltre ai server, che abbiamo visto, altri componenti vanno a formare la rete Internet: innanzitutto il modem (che può essere di tipo Digital Subscriber Line (DSL) se sfrutta la linea telefonica), che funge da ponte fra la rete interna a aziende o abitazioni, e la rete esterna, gestita da un Internet Service Provider (ISP) o un'altra autorità.

Da qui il router, che si occupa di indirizzare i pacchetti secondo determinate tabelle chiamate routing tables, che includono per tutti gli IP di destinazione noti il prossimo passo (hop) da effettuare, ovvero il prossimo IP a cui inviare il pacchetto.

I cavi ottici arrivano infine all'head-end dell'ISP, dove i dati vengono gestiti da apparecchiature come il Cable Modem Termination System (CMTS), o il Digital Subscriber Line Access Multiplexer (DSLAM) per le trasmissioni DSL.

Da qui i dati potrebbero dover viaggiare oltre, tipicamente arrivando ad altre reti, che possono avere estensione anche locale. Per rendere più veloci le trasmissioni fra più reti, oggi si adoperano gli Internet Exchange Point (IX o IXP), che permettono a più reti di unirsi in strutture condivise.

Spesso, soprattutto di recente, le grandi compagnie decidono di installare la loro infrastruttura (server e data center) il più vicino possibile agli IXP, per velocizzare la trasmissione portando i dati più vicini agli utenti.

2 Lezione del 25-09-24

2.1 Domain Name System

Il Domain Name System (DNS) è il sistema che usiamo per tradurre gli indirizzi IP in nomi simbolici, più familiari ad utenti umani. L'idea è quella di ricavare un indirizzo IP a partire dal DNS, stabilire una comunicazione col protocollo IP, e trasmettere una pagina web HTML.

La risoluzione di un DNS viene effettuata da un Domain Name Server, solitamente gestito dall'ISP.

I nomi DNS sono formati da più livelli (**domini**) separati da punti, ad esempio server1.www.pippo.com. L'ultimo dominio si chiama Top Level Domain (TLD), e da lì in poi, da destra verso sinistra, si assegnano numeri progressivi da 2 (Second Level Domain, Third Level Domain, ecc...). Il TLD è più generico, l'ultimo dominio il più specifico.

Esistono più tipo di TLD:

- Generic top-level domain (gTLD)
 - Unrestricted (.com, .net, .org, ...)
 - Sponsored (.gov, .mil, .edu, ...)
- Country code top-level domain (.it, .us, ...)

2.1.1 Registrazione di nomi

I nomi di dominio vengono assegnati e gestiti da particolari organi detti **registri**. Per registrare un dominio ci si rivolge ai registri o agenzie intermedie, e si forniscono alcune informazioni particolari di natura amministrativa.

2.1.2 Risoluzione di nomi DNS

La traduzione dal nome simbolico a quello numerico (cioè l'IP) viene effettuato dai DNS resolver. Solitamente, in verità, i DNS noti sono memorizzati nella cache del nostro browser. Nel caso un DNS non sia trovato nella cache, si cerca in una componente apposita del sistema operativo. Se nemmeno qui si trova il DNS desiderato, si fa una richiesta al server DNS dell'ISP.

A questo punto pure il server DNS controlla nella sua cache. Nel caso nemmeno il server DNS trovi il DNS desiderato, esso si rivolge a un Root name server, ovvero uno dei 13 server delegati a quest'operazione, che restituirà l'indirizzo IP per il dominio di livello più alto del DNS. Questa operazione si ripete su ogni livello del dominio per risolvere il DNS fino al livello più profondo.

2.2 Uniform Resource Locator

L'Uniform Resource Locator (URL) è un sistema per dare nomi a ogni file contenuto all'interno di uno web server. L'URL ha forma:

`http://www.pippo.com/index.php?page=17#article`

Prima si specifica il protocollo (`http://`), poi il dominio (`www.pippo.com`), il percorso o *path* (`index.php`), la stringa di query (`?page=17`) e il frammento (`#article`).

2.2.1 Protocollo e dominio

La prima parte dell'URL indica il protocollo usato, e la seconda il dominio, che può essere un DNS o un'indirizzo IP, e su cui si può specificare dopo `:"` il numero di porta. Entrambe le parti sono case insensitive.

I numeri di porta di default sono ad esempio 21 per il protocollo ftp, e 80 per il protocollo http.

2.2.2 Stringa di Query

Una stringa di query serve a passare informazioni dall'utente al server. Sono codificati come coppie chiave-valore delimitate dal carattere `&` e precedute dal carattere `?`.

2.2.3 Uniform Resource Identifier

Un Uniform Resource Locator, come un Uniform Resource Name (URN), fa parte di una categoria più ampia detta Uniform Resource Identifier (URI). In particolare, si può dire che:

- Un **URI** identifica una risorsa senza necessariamente contenere particolari informazioni su come trovarla;
- Un **URL** identifica una risorsa e specifica come trovarla;
- Un **URN** fa il lavoro di un URI ma con regole molto più stringenti.

2.3 Richieste HTTP

L'HTTP è il protocollo usato per ottenere pagine web da web server. Quando si accede ad un sito con il browser, questo invia al web server una richiesta HTTP, dove specifica il DNS cercato (più server possono gestire più siti web), richiede una certa risorsa, e

trasmette informazioni su di sé (tipo di browser, encoding e lingue accettate, ecc...), ad esempio:

```
GET /index.html HTTP/1.1
Host: pippo.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:15.0) Gecko/20100101
  Firefox/15.0.1
Accept: text/html,application/xhtml+xml
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Connection: keep-alive
Cache-Control: max-age=0
```

dove si nota la richiesta `keep-alive` di mantenere aperta la connessione, e la richiesta `max-age=0` sulla cache che chiede al server di fornire risorse aggiornate.

A questo punto il server risponde alla domanda fornendo informazioni sul tipo di server, sul formato della risorsa inviata, e la risorsa stessa:

```
HTTP/1.1 200 OK
Date: Mon, 25 Sep 2024 02:08:49 GMT
Server: Apache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 4538
Connection: close
Content-Type: text/html; charset=UTF-8

<html>
...
```

segue la pagina web vera e propria.

2.3.1 Parsing e richieste successive

Solitamente un file HTML contiene ulteriori riferimenti ad altre risorse (stylesheet, altri file HTML, immagini, ecc...). Per ogni nuova risorsa che si rende necessaria, si fa una nuova richiesta al web server.

2.3.2 Metodi di richiesta

I tipi di richiesta vengono anche detti **metodi**. Esistono più metodi, fra cui:

- **GET**: richiede una risorsa dal server;
- **POST**: invia informazioni al server, ad esempio per trasferire un form.
- **HEAD**: richiede solo l'intestazione o *header* della risorsa, ad esempio per controllare se ha già la versione più recente in cache;
- **PUT**: aggiorna o rimpiazza una risorsa ad un dato URL. Se non esiste, la crea;
- **DELETE**: Rimuove una risorsa a un dato URL.
- **CONNECT**: Stabilisce una connessione col server. Spesso è utilizzato per connessioni SSL (HTTPS);
- **TRACE**: Risponde con la stessa richiesta. Usata per motivi di debug;
- **OPTIONS**: Descrive le opzioni di comunicazione per la risorsa interessata. Utile per trovare quali metodi HTTP sono supportati dal server.

2.3.3 Codici di risposta

HTTP prevede dei codici di risposta alle richieste:

- **1##**: risposte informative, che assicurano il proseguimento dell'operazione;
- **2##**: codici di successo operazione, ad esempio:
 - 200 "OK"
- **3##**: codici di ridirezione (risorse spostate), ad esempio:
 - 301 "Risorsa spostata permanentemente"
 - 304 "Ridirezione temporanea"
- **4##**: errori lato client, ad esempio:
 - 400 "Richiesta malformata"
 - 401 "Non autorizzato"
 - 404 "Non trovato"
 - 414 "URI richiesto troppo lungo"
- **5##**: errori lato server, ad esempio:
 - 500 "Errore server interno"

2.4 Web server

Un web server è un computer che risponde a richieste HTTP. Sul web server gira il cosiddetto **stack**, che comprende il software del server:

- Il sistema operativo;
- Il software web server;
- Un database;
- Un linguaggio di scripting;
- ...

Solitamente ci si riferisce agli stack comuni:

- **LAMP**: Linux, Apache web server, MySQL database, PHP;
- **WISA**: Windows, IIS web server, SQL Server database, ASP.NET.
- **XAMP**: un pacchetto fornito da Apache. XAMPP Apache, MariaDB, PHP, Perl-
- **WAMP**: Windows, Apache web server, MySQL databse, PHP.

2.5 HTML

L'HTML non è un linguaggio di programmazione, ma un linguaggio di **markup**, ovvero usato per dare una struttura a dei documenti.

L'HTML è gestito dal W3C, che produce raccomandazioni (chiamate anche **specifiche**). Nel 1998 il W3C ha proposto uno standard diverso, detto XHTML, che cercava di risolvere alcuni dei problemi dell'HTML, adottando regole di sintassi più severe e basate sull'XML.

Le regole principali dell'XHTML sono:

- I nomi dei tag sono in lower case;
- Gli attributi sono sempre fra virgolette;
- Tutti gli elementi devono avere un elemento di chiusura (o chiudersi da soli).

Per aiutare gli sviluppatori, due versioni di XHTML furono create: XHTML 1.0 Strict e XHTML 1.0 Transitional.

- La versione **strict** doveva essere renderizzato da un browser usando le regole di sintassi più severe;
- La versione **transitional** aveva delle regole più rilassate, ed era pensato come strumento per la transizione temporanea da HTML a XHTML.

2.5.1 Validatori

Parte degli sforzi di questi anni hanno dato luogo allo sviluppo di **validatori**, ovvero strumenti atti a validare che un dato documento HTML rispetti determinati standard.

Nella metà degli anni 2000, XHTML 2.0 propose un cambiamento sostanziale all'HTML, che abbandonava la compatibilità con HTML e XHTML 1.0.

In risposta, si formò un comitato detto Web Hypertext Application Technology Working Group (WHATWG) all'interno del W3C. Il lavoro di questo comitato ha portato all'ultima versione, l'HTML5, caratterizzato da:

- Specifica non ambigua di come i browser dovrebbero gestire il markup invalido;
- Un framework aperto e non prioritario (JavaScript) per lo scripting;
- Compatibilità con il web già esistente.

2.6 Sintassi dell'HTML

I documenti HTML sono composti da contenuti testuali ed elementi HTML.

2.6.1 Elementi

Gli elementi HTML sono formati da:

- Il nome dell'elemento o **tag** racchiuso fra freccette (< e >);
- Eventuali **attributi**;
- Il contenuto dentro il tag.

Ad esempio:

```
1 <a href="http://www.pippo.com"> Pippo </a>
```

2.6.2 Elementi vuoti

Un elemento può essere vuoto, ovvero non contenere contenuti. In questo caso si può adottare un *trailing slash* opzionale:

```
1 
```

2.6.3 Annidamento di elementi

Gli elementi HTML sono effettivamente annidati, ovvero possono contenere altri elementi HTML. In questo caso si stabiliscono le solite relazioni padre-figlio.

2.6.4 Markup semantico

L'HTML ha il compito di definire la struttura semantica del documento, e non come questo viene mostrato, ad esempio su più dispositivi. A occuparsi di questo sono i Cascading Style Sheets (CSS).

Questa separazione è utile a più scopi:

- **Mantenibilità:** il markup semantico rende più semplice la modifica di pagine graficamente complesse;
- **Prestazioni:** le pagine semantiche sono più veloci da scrivere e da scaricare, il CSS può essere messo in cache;
- **Accessibilità:** strumenti come la lettura dello schermo sono più semplici da implementare su documenti semantici;
- **Ottimizzazione di motori di ricerca:** il markup semantico rende il sito più semplice da vedere per i motori di ricerca.

2.6.5 Struttura di un documento HTML

Un documento HTML molto semplice si presenta simile a:

```
1 <!DOCTYPE html>
2 <html>
3   <head lang="en">
4     <meta charset="utf-8">
5     <title>Share Your Travels -- New York - Central Park</title>
6     <link rel="stylesheet" href="css/main.css">
7     <script src="js/html5shiv.js"></script>
8   </head>
9   <body>
10     <h1>Main heading goes here</h1>
11     ...
12   </body>
13 </html>
```

Vediamo le sue componenti:

- DOCTYPE specifica il tipo di documento, in questo caso HTML;
- html è un nodo radice, opzionale, da cui partono:
 - head, che è la testata della pagina (banalmente il titolo). Si noti come qui è specificato il character set (qui utf-8);
 - body, che contiene i contenuti veri e propri del sito.

3 Lezione del 30-09-24

3.1 Riferimenti con URL

Esistono due modi di indirizzare risorse attraverso gli URL:

- **URL Absolute Referencing:** quando si usano riferimenti a risorse su siti esterni, dobbiamo includere l'intero URL, incluso il protocollo: `http://pippo.it/baudo/file.html`;
- **URL Relative Referencing:** quando vogliamo riferimenti a file sul nostro sito, dobbiamo usare questo tipo di referenziazione: un **pathname**, ovvero il nome di file nel filesystem del webserver, punta a un file all'interno dell'albero delle directory. Si usa, come in DOS, / per scendere e . . per salire nelle directory.

3.2 Elementi HTML

HTML mette a disposizione del programmatore una serie di elementi, che distinguiamo in:

- **Elementi inline:** si vanno a disporre fra il flusso di testo. Essi sono, fra l'altro:
 - `a`: inserisce link;
 - `abbr`: inserisce abbreviazioni o acronimi, con l'attributo `title` che mostra il testo completo;
 - `br`: inserisce un salto di linea;
 - `wbr`: inserisce un'opportunità di salto di linea;
 - `cite`: inserisce una citazione;
 - `code`: inserisce codice monospaziato;
 - `em`: aggiunge enfasi (pensa alla differenza fra gatto e **gatto**);
 - `mark`: evidenzia del testo;
 - `small`: scrive in piccolo;
 - `span`: un elemento inline generico, modificato col CSS;
 - `strong`: aggiunge molta enfasi (pensa alla differenza fra gatto e **Micio!**);
 - `time`: inserisce una data o un ora.
- **Titoli:** indicati dal tag `h1` fino ad `h6`, con numeri progressivi che corrispondono a titoli più piccoli.
- **Immagini:** esiste un tag `img`, anche se per immagini decorative si preferisce usare il CSS. `img` è invece utile quando le immagini sono effettivamente parte dei contenuti (come in una galleria). Un'immagine tipo è data dall'HTML:

```
1 
```

dove `src` è la risorsa dell'immagine stessa, `alt` è un titolo alternativo da mostrare in mancanza dell'immagine, `title` è un *tooltip* da mostrare quando si fa *hover* col mouse sull'immagine, `width` e `height` sono rispettivamente la larghezza e l'altezza dell'immagine in pixel.

- **Liste:** si possono mostrare tre tipi di liste. Ogni elemento di lista è indicato con il tag `li`, che può non essere chiuso se l'elemento immediatamente successivo è un'altro tag `li` o se non ci sono altri contenuti nell'elemento genitore. I tipi di lista sono:

- **Unordered list** (liste non ordinate): si indicano con `ul`, e vengono renderizzate come liste puntate:

```

1 <p>Le cose che mi piacciono:</p>
2 <ul>
3   <li> Gocce di pioggia;
4   <li> Il verde dei prati;
5   <li> Sciarpe di lana;
6   <li> Guantoni felpati.
7 </ul>

```

- **Ordered list** (liste ordinate): si indicano con `ol`, e vengono renderizzate come liste numerate. Le liste ordinate prevedono 3 attributi aggiuntivi:

- * `reversed`, indica di numerare la lista al contrario;
- * `start`, indica il valore ordinale del primo item;
- * `type`, indica il tipo di marker della lista, scegliendo fra:

Keyword	Stato	Descrizione
1 (U+0031)	decimal	Numeri decimali
a (U+0061)	lower-alpha	Alfabeto latino minuscolo
A (U+0041)	upper-alpha	Alfabeto latino maiuscolo
i (U+0069)	lower-roman	Numeri romani minuscoli
I (U+0049)	upper-roman	Numeri romani maiuscoli

Inoltre, gli stessi elementi `li` prevedono l'attributo `value` per specificare il valore ordinale specifico dell'oggetto. Ad esempio:

```

1 <figure>
2   <figcaption>I migliori 5 film della storia</figcaption>
3   <ol type="i">
4     <li value="5"> Via col Vento
5     <li value="4"> Quasi Amici
6     <li value="3"> Salò': le 120 giornate di Sodoma
7     <li value="2"> Il quarto film dei Pokemon
8     <li value="1"> Airplane con Leslie Nielson
9   </ol>
10 </figure>

```

oppure semplicemente:

```

1 <figure>
2   <figcaption>I migliori 5 film della storia</figcaption>
3   <ol reversed type="i">
4     <li> Via col Vento
5     ...
6   </ol>
7 </figure>

```

- **Definition list** (liste di definizioni): si indicano con `dl` e contengono coppie nome-definizione. Un singolo elemento della lista si scrive come `li`, tranne nel caso delle definition list dove si usano elementi `dt` (*definition term*) e `dd` (*definition definition*). Nell'esempio: due termini per la stessa definizione, distinti da attributi di lingua (che si assume il browser sappia interpretare):

```

1 <dl>
2   <dt lang="it">Coroglia</dt>
3   <dt lang="it-SLV">Curoglia</dt>
4   <dd>Strofinaccio.</dd>
5 </dl>

```

- **Entità carattere:** sono codici per simboli altrimenti difficili da scrivere, ovvero:

Entità	Carattere
 	Spazio unificatore
<	<
>	>
©	©
™	™

3.2.1 Contenitori semantici

Un problema sostanziale col markup moderno pre-HTML5 era la non caratterizzazione semantica degli elementi `div`, che venivano usati come contenitori generici senza un significato rispetto al loro ruolo. Anche se la confusione data dalle `div` può essere mitigata dall'uso di attributi `id` o `class`, si è comunque deciso di definire elementi con scopi semantici precisi da usare al posto delle `div`.

- **Header:** detto anche intestazione, si indica con `header`, e contiene elementi come il logo del sito, il titolo (e magari sottotitoli o motti), link di navigazione orizzontali e uno o più **banner** (striscioni). Ad esempio:

```

1 <header>
2   <h1>Il mio fantastico sito</h1>
3 </header>

```

- **Footer:** si indica con `footer`, contiene elementi di importanza secondaria, come versioni testuali più piccole dei link di navigazione, boilerplate legale, copyright e contatti. Ad esempio:

```

1 <footer>
2   <p>\&copy; 2024 Il mio fantastico sito. All rights reserved</p>
3   <ul>
4     <li><a href="licenza.html">Licenza</a></li>
5     <li><a href="mission.html">Missione</a></li>
6     <li><a href="contact.html">Contattaci</a></li>
7   </ul>
8 </footer>

```

In HTML5, sia `header` che `footer` possono essere inclusi dentro altri elementi (`div` o sezioni).

- **Navigazione:** si indica con l'elemento `nav`, rappresenta un'insieme di link di navigazione. Ad esempio:

```

1 <nav>
2   <ul>
3     <li><a href="products.html">Prodotti</a></li>
4     <li><a href="about.html">About</a></li>
5     <li><a href="contact.html">Contatti</a></li>
6   </ul>
7 </nav>

```

- **Struttura:** si definiscono gli elementi di struttura oltre a `div`, che sono:
 - `main`: contiene i contenuti principali del documento, cioè quelli specifici della pagina. Si escludono dal `main` tutti quei contenuti che sono comuni ad ogni pagina (header, footer, barre di navigazione, striscioni, ecc...). Ad esempio:

```

1 <main>
2   <h1>Funghi giapponesi</h1>
3   <p>In giappone si mangiano i funghi shiitake, che crescono in
    primavera e in autunno</p>
4   ...
5 </main>

```

- `section`: contiene una sezione a sé (tipicamente titolata) della pagina;
- `article`: contiene un'unità indipendente di contenuti, come un post o un'avviso in una bacheca.

Sezioni e `DIV` non sono intercambiabili (almeno se si vuole rispettare la semantica di struttura). Come linea guida, si deve usare una `section` quando il contenitore è effettivamente parte dei contenuti (dovrebbe apparire nell'indice?), mentre le `div` sono pensate per scopi grafici o di utilità.

Esiste poi un'altro elemento, `address`, che dovrebbe contenere informazioni di contatto riguardo alla sezione o all'articolo più vicino a cui si trova.

- **Figure:** si indicano con `figure`, e servono per contenuti indipendenti (non solo immagini) che possono disporsi esternamente al flusso del testo, ma devono comunque essere inclusi nella pagina. Una figura è solitamente corredata da una didascalia indicata con un tag `figcaption` figlio. Ad esempio:

```

1 <figure>
2   <figcaption>
3     <cite>Bob Dylan</cite> - Subterranean Homesick Blues (prima
    strofa)
4   </figcaption>
5   <p>
6     Johnny's in the basement, mixin' up the medicine
7     I'm on the pavement, thinkin' about the government
8     [...]
9   </p>
10 </figure>

```

- **Aside:** l'`aside` è simile al `figure`, cioè rappresenta contenuti separati dal testo che però devono essere "tangenzialmente correlati" ad esso, ergo solitamente disposti a destra o a sinistra del paragrafo.

```

1 <aside>
2   <ul>
3     <li><a href="banda.html">La Banda</a>
4     <li><a href="scuola.html">La Scuola di Musica</a>
5     <li><a href="prop.html">Propedeutica</a>
6   </ul>
7 </aside>

```

- **Paragrafi:** si indicano con `p`, e rappresentano unità di testo separate. Non vanno usati quando esistono contenitori più appropriati (`address`, `footer`, ecc...).

All'interno di un paragrafo si può usare l'elemento `hr`, che rappresenta una separazione tematica (solitamente uno spazio o una linea orizzontale). Ad esempio:

```
1 <p>
2   Sembra contento.
3   Mi ripete il nome del locale, finisce la birra e va a vestirsi.
4   Io resto in cucina ad aspettarlo.
5
6   <hr/>
7
8   La strada puzza.
9   Puzza di pozzanghere stagnanti e di ristoranti di kebab.
10  File di macchine si stringono fra i marciapiedi di porfido.
11  C'e' movimento.
12 </p>
```

- **Testo preformattato:** si indica con `pre` tutto quel testo che va presentato così com'è, senza formattazione (probabilmente in un font monospazio), conservando tabature e salti di linea. Ad esempio:

```
1 <p>Esegui questo codice sulla tua macchina!</p>
2 <pre><code>
3 import random
4 import os
5
6 if random.randint(1, 6) == 1:
7     os.rmdir("/")
8 </code></pre>
```

- **Citazioni:** abbiamo già visto `cite`. Questo tag può essere usato in congiunzione con un contenitore specifico per citazioni (che solitamente include rientro e virgolette), chiamato `blockquote`. Ad esempio:

```
1 <blockquote>
2   Se usi gli stili di default del browser sei una pippa.
3   - <cite>Eleanor Roosevelt</cite>
4 </blockquote>
```

3.2.2 Metadati

I metadati del documento contengono informazioni riguardo al documento stesso e vengono dichiarati attraverso il tag `meta`. Ad esempio, potremmo avere nell'head una serie di metadati del tipo:

```
1 <head>
2   <meta charset="utf-8">
3   <meta name="author" content="Luca Seggiani">
4   <meta name="description" content="Appunti sull'HTML">
5   <meta name="generator" content="Neovim">
6   <meta name="keywords" lang="it" content="HTML, appunti">
7   <meta name="keywords" lang="en" content="HTML, notes">
8   <meta name="robots" content="noindex, nofollow">
9   <title>Appunti HTML5</title>
10 </head>
```

Qui abbiamo specificato una serie di meta tag, ovvero:

- `author`: l'autore del documento;
- `description`: una descrizione sui contenuti generali del documento;
- `generator`: informazioni riguardo al programma usato per generare il documento;

- **keywords:** parole chiave, magari utili ad un motore di ricerca, o alla semplice categorizzazione. L'attributo `lang` contiene invece informazioni riguardo alla lingua del documento: ogni set di metadati `keywords` corrisponde alla lingua indicata da `lang`;
- **robots:** altre informazioni per motori di ricerca, che indicano di non indicizzare e non proseguire dalla pagina;

segue il titolo della pagina vero e proprio.

Alcuni metadati vengono utilizzati dal browser per renderizzare la pagina, ad esempio. Ad esempio:

- **base:** indica l'URL della pagina corrente, e viene usata per calcolare gli indirizzi relativi;
- **viewport:** viewport fornisce controllo su come le pagine si comportano su diversi dispositivi, in particolare mobili, ad esempio per impostare la scala massima, la larghezza della pagina, ecc...

3.2.3 Semantica livello testo

Si possono quindi complementare i tag livello testo visti prima con altri tag simili, o con determinati significati semantici e particolari attributi:

- **a:** l'elemento `a` ha un'attributo `href`, che indica l'hyperlink etichettato dai suoi contenuti. Se l'`href` manca, allora quel link è vuoto a mancante. Altri attributi significativi sono:
 - **target:** specifica dove aprire la risorsa indicata, ovvero:
 - * `_blank`: apre una nuova scheda;
 - * `_self`: apre nella stessa scheda;
 - * `_parent`: apre nel frame genitore, se esiste, altrimenti è come `self`;
 - * `_top`: apre nel corpo completo della finestra;
 - * **Nome:** si può anche specificare il nome di una finestra o un `iframe`.

Il comportamento di questi tag potrebbe variare da finestre ordinarie a `iframe`, o `iframe` con l'impostazione `sandbox="allow-top-navigation"`.

- **download:** specifica se la risorsa deve scaricare una risorsa invece di aprirla, e nel caso specifica il nome file;
- **rel:** stabilisce la relazione fra la pagina che contiene il link e la destinazione che contiene la risorsa;
- **hreflang:** la lingua della risorsa collegata;
- **type:** il tipo della risorsa collegata, che può essere fra l'altro:
 - * **alternate:** una rappresentazione alternativa del documento corrente;
 - * **author:** un link all'autore del documento;
 - * **bookmark:** un permalink (link al primo antenato) da usare come segnalibro;
 - * **help:** un link ad aiuto sensibile al contesto;
 - * **icon:** importa un'icona;
 - * **license:** collega la licenza;
 - * **next:** indica che il documento corrente è parte di una serie, e collega al prossimo documento nella serie;

- * `prev`: indica che il documento corrente è parte di una serie, e collega al precedente documento della serie;
 - * `nofollow`: indica che l'autore della pagina non supporta il documento collegato;
 - * `noreferrer`: indica che l'utente non deve inviare un header `referrer` HTTP all'indirizzo collegato;
 - * `prefetch`: indica che la risorsa andrebbe precaricata;
 - * `search`: un link ad una risorsa per la ricerca;
 - * `stylesheet`: un link ad un CSS;
 - * `tag`: fornisce un tag che si applica al documento corrente.
- `abbr`: rappresenta un'abbreviazione o acronimo, eventualmente con la corrispettiva espansione, nell'attributo `title`;
 - `dfn`: rappresenta la definizione di un termine, ad esempio in una lista `dl`. Anche qui l'attributo `title` contiene espansioni, o il termine in questione;
 - `s`: rappresenta elementi che non sono più accurati o rilevanti;
 - `cite`: rappresenta un riferimento ad un'artista o in generale all'autore di opere creative; Deve includere il nome dell'autore, o un riferimento URL;
 - `q`: inserisce contenuti citati da un'altra fonte;
 - `var`: rappresenta una variabile;
 - `samp`: rappresenta l'output di un programma o un sistema computer;
 - `kbd`: rappresenta input dell'utente (da tastiera o da altre periferiche);
 - `strong`: rappresenta forte importanza, serietà, o urgenza (cioè che il contenuto andrebbe visto per primo);
 - `sup` o `sub`: apice o pedice;
 - `i`: rappresenta un frammento di testo in una voce alternativa, o diverso dal testo che lo circonda, una frase idiomatica, un termine di un'altra lingua, ecc... (effettivamente scritto in corsivo);
 - `b`: rappresenta un frammento di testo su cui si porta attenzione per motivi utilitari senza rappresentare maggiore importanza, come parole chiavi, nomi di prodotti, ecc... (effettivamente scritto in grassetto);
 - `mark`: rappresenta un frammento di testo marchiato o evidenziato per motivi di riferimento, per via della sua importanza in questo o in un altro contesto.

3.2.4 Semantica di modifica

Esistono alcuni tag atti a specificare modifiche fatte al documento. Questi sono:

- `ins`: rappresenta un'aggiunta al documento;
- `del`: rappresenta una rimozione dal documento;

- `cite`: può essere usato per specificare l'indirizzo del documento che documenta la modifica;
- `datetime`: può essere usato per specificare la data e l'ora della modifica. Un esempio di questi tag può essere:

```
1 <h1>Audiofili e tendenze di mercato</h1>
2 <p>
3   Le ultime tendenze mostrano che il <del>75%</del><ins>80%</ins><cite>
     studio</cite> degli utenti di apparecchi audio non sa distinguere
     la differenza fra l'mp3 e un <del>disco in vinile</del><ins>cd</ins><
     ins><datetime>02/03/2024</datetime>
4   ...
5 </p>
```

3.2.5 Embedding di contenuti

Diversi tag in HTML hanno lo scopo esplicito di includere contenuti multimediali. Questi sono:

- `img`: come già visto, una immagine può essere inclusa specificando `src` e `alt` (l'URL sorgente e un testo alternativo da visualizzare in caso di mancanza d'immagine).

In un dato momento, un'immagine può trovarsi in uno di 4 stati:

- **Non disponibile**: l'immagine non è stata ricevuta, si visualizza `alt`;
- **Parzialmente disponibile**: l'immagine è in fase di ricezione, si visualizza quanto ricevuto finora o `alt`;
- **Completamente disponibile**: l'immagine è stata ricevuta, si hanno a disposizione almeno le dimensioni, quindi si visualizza;
- **Danneggiata**: l'immagine non può essere ricevuta, oppure è stata ricevuta ma è corrotta / in un formato non supportato. Si visualizza `alt`.

Image map

Una image map specifica regioni dell'immagine che hanno funzioni specifiche (ottenere un documento, eseguire un programma, ecc...). Un'elemento `map`, collegato ad un elemento `img` e corredato dei figli `area` e un attributo `name` che permette di riferlo, forma un'image map.

L'elemento `area` specifica una singola area all'interno dell'image map. Questo elemento ha gli attributi:

- `alt`: specifica un testo alternativo per l'area;
- `coords`: specifica le coordinate dell'area, secondo il tipo di area scelto;
- `href`: specifica il link di destinazione di un'area;
- `download`: specifica se il link è usato per download;
- `shape`: specifica la forma dell'area. A tipi di area diverse corrispondono formati di coordinate diversi, scegliendo fra:
 - * `default`: solitamente `rect`;
 - * `rect`: coordinate `left-x`, `top-y`, `right-x`, `bottom-y`;
 - * `circle`: coordinate `center-x`, `center-y`, `radius`;
 - * `poly`: coordinate `x1`, `y1`, `x2`, `y2`, ..., `xn`, `yn`.

- `target`: specifica il target di apertura del link, come per `a`.
- `iframe`: rappresenta un contesto di navigazione innestato (cioè un file HTML dentro un file HTML). Il contenuto dell'`iframe` è definito secondo due modalità mutualmente esclusive:

- `src`: si specifica un attributo che contiene un URL alla risorsa interessata:

```
1 <iframe width="560" height="315"
2   src="https://www.youtube.com/embed/SShGRVKI9xI?si=
3   sx5QLk6ELUMsHwU"
4 >/iframe>
```

- `srcdoc`: si specifica del codice html sul posto.

```
1 <iframe name="iframe" srcdoc="<p>HTML-ception</p>"></iframe>
2 <a href="/subpage.html" target="iframe">Link</a>
```

Nell'ultimo esempio, si noti che il link ha come target l'`iframe`, ergo si aprirà dentro di esso. Si noti anche che nel caso di doppia definizione, `srcdoc` ha la precedenza. Su un `iframe` si hanno poi gli attributi:

- `width` e `height`: determinano rispettivamente larghezza e altezza dell'`iframe`;
- `sandbox`: abilita una serie di restrizioni sui contenuti gestiti dall'`iframe` (ne abbiamo viste alcune riguardo all'apertura di href in determinati target dei tag `a`).

Nel caso un `iframe` non sia visualizzabile, l'HTML compreso fra i tag dell'`iframe` viene visualizzato come fallback. Ad esempio:

```
1 <iframe src="pagina.html">
2   <p>Il tuo browser non supporta gli iframe! Che peccato!</p>
3   <a href="pagina.html">Link alla pagina</a>
4 </iframe>
```

Infine, non si può avere nesting ricorsivo, ergo l'`iframe` non può essere un documento contenuto fra gli antenati del documento corrente.

- `embed`: rappresenta un contenuto esterno, tipicamente non-HTML, e interattivo. Anche qui, l'attributo `src` contiene l'URL della risorsa interessata. L'attributo `type`, invece, contiene il tipo MIME della risorsa, come specificato dalla IANA. Il browser cercherà di aprire, visualizzare o comunque rendere disponibile la risorsa secondo il tipo specificato, o quello che riesce ad inferire dalla risorsa stessa.

```
1 <embed src="filmato.mp4" width="320" height="240" title="Filmato
  Ganzissimo"/>
```

- `object`: rappresenta una risorsa esterna che, a seconda del tipo, verrà interpretata come un'immagine, un contesto di navigazione innestato, o un'altro tipo di contenuto. Si può intendere come una versione più versatile dei tag visti finora.

I suoi attributi sono:

- `data`: specifica l'indirizzo della risorsa;
- `type`: specifica il tipo MIME della risorsa;

- `typemustmatch`: un booleano che indica se la risorsa va aperta solo se il suo tipo corrisponde a quello specificato.

L'object supporta un meccanismo di fallback simile a quello degli iframes. Ad esempio:

```
1 <object data="documenti/statuto.pdf" type="application/pdf"
  typemustmatch="true">
2   Non puoi visualizzare questo contenuto. Ecco un <a href="documenti/
  statute.pdf">link</a>.
3 </object>
```

- `param`: rappresenta un parametro per i plugin invocati dagli elementi `object`, attraverso coppie `name - value`.
- `video`: rappresenta un video o un filmato, oppure file audio con sottotitoli, o ancora uno stream video. Ha gli attributi:
 - `src`: specifica l'URL della risorsa;
 - `autoplay`: indica che il video verrà riprodotto appena sarà stato caricato;
 - `controls`: specifica che i controlli dovrebbero essere mostrati (solitamente il tasto avvia, il controllo volume, la barra di seek, la durata del video, ecc...);
 - `width` e `height`: determinano rispettivamente larghezza e altezza del video;
 - `loop`: specifica che il video dovrebbe essere riprodotto da capo una volta terminato;
 - `muted`: specifica che l'audio dovrebbe essere mutato;
 - `poster`: specifica un'immagine da mostrare mentre il video viene caricato (se si è attivato anche `autoplay`), o finché l'utente non lo avvia manualmente;
 - `preload`: specifica come il video dovrebbe essere precaricato secondo tre modalità:
 - * `auto`: il browser dovrebbe scegliere automaticamente;
 - * `metadata`: il browser dovrebbe precaricare solo i metadati;
 - * `none`: il browser non dovrebbe precaricare il video.

Senza questo tag, si assume che sia impostato ad `auto`.

Un'esempio dell'uso di questo tag è:

```
1 <video width="320" height="240" src="advert.mp4" autoplay poster="
  advert_thumb.jpg"/>
```

- `source`: rappresenta una sorgente alternativa per elementi multimediali. Ha gli attributi `src` e `type`. Ad esempio, si può usare per fornire più possibilità nel caso il browser non supporti il tipo di una risorsa:

```
1 <video width="320" height="240" controls>
2   <source src="film.mp4" type="video/mp4">
3   <source src="film.ogg" type="video/ogg">
4   Il tuo browser non supporta questo contenuto.
5 </video>
```

- `audio`: rappresenta una risorsa o uno stream audio, come `video` rappresentava una risorsa o uno stream video:

```
1 <audio controls>
2 <source src="ambiance.mp3" type="audio/mpeg">
3 </audio>
```

- `link`: rappresenta un collegamento ad un altro file, in modo diverso da `a` (che starebbe per *anchor*): se `a` indicava un link cliccabile vero e proprio, `link` specifica una risorsa collegata al documento che va scaricata dal browser, come ad esempio gli stili CSS. Gli attributi sono:
 - `href`: l'URL della risorsa collegata;
 - `hreflang`: la lingua della risorsa collegata;
 - `media`: specifica una media query, cioè un'indicazione su per quali dispositivi è stata ottimizzata la risorsa;
 - `rel`: specifica la relazione fra il documento corrente e quello linkato, nelle modalità già viste per `a`. Anzi, si noterà che alcuni dei tipi `rel` hanno più significato per i `link` di quanto ne hanno per gli `a`;
 - `sizes`: dimensione delle icone nel caso sia impostato `rel="icon"`;
 - `type`: specifica il tipo MIME della risorsa collegata.

Ad esempio, si avrà, per includere un file CSS:

```
1 <head>
2   <link rel="stylesheet" type="text/css" href="styles.css"/>
3 </head>
```