Storia del www
gli ipertesti

- vannevar bush idealizza il memex:
  - basato su un meccanismo di fotografia e microfilmazione automatica di fogli, sulla possibilità di punzonare secondo codici prestabiliti i microfilm e su un meccanismo di ricerca rapida di microfilm sulla base di queste punzonature
- theodore nelson e xanadu:
  - basato su server che costruivano documenti virtuali basati su pezzi di testi di lunghezza arbitraria e client che permettevano creazione di link, modifica dei documenti e che possedevano un sistema efficace di gestione e pagamento per i documenti
- douglas englebart e augment:
  - sistema di videoconferenza, editing di testi gerarchici (outline processor) e ipertestuali e di supporto per il lavoro cooperativo
  - dotato di interfaccia a fineste, mouse e altri meccanismi rivoluzionari di I/O
- bill atkinson e hypercard:
  - software di produttività personale
  - permetteva a chiunque di organizzare testi, immagini, piccoli database e programmi attraverso anche un linguaggio di scripting molto evoluto (hypertalk)
  - permetteva di realizzare quindi applicazioni più potenti e facili

il world wide web

- gli inizi:
  - 1989 – un gruppo di ricercatori del CERN viene incaricato della realizzazione di un meccanismo per la diffusione rapida di articoli, appunti e opinioni
  - 1991 – primo prototipo realizzato in client-server su architettura NeXT
  - 1992 – il NCSA (national centre for supercomputing applications) decide di realizzare una sua versione del WWW: il server NCSA e il browser Mosaic
  - in poco tempo l'NCSA aveva deciso di non spendersi più sul www
  - 1993 – la Mosaic corporation (poi Netscape) viene fondata
  - 1994 – netscape navigator
- world wide web consortium (W3C):
  - 1994 – per promuovere e uniformare lo sviluppo delle tecnologie del web
  - sviluppo dei più importanti standard del web (URI, HTTP, CSS)
  - dirigenza del gruppo di lavoro su XML e standard collegati
    identificò tecnologie alternative alla censura centralizzata (PICS, che si evolverà nelle tecnologie del parental control)
- la prima guerra dei browser:
  - microsoft realizza il browser internet explorer e il server microsoft information server
  - per evitare la nascita di due internet separate, viene fondato il W3C con fondi di ricerca e università
  - 1997 – internet explorer è veloce, compatibile e stabile
- dominanza di explorer:
  - 1998 – netscape ammette la disfatta (nel 1999 condivide l'ultima versione 4.7), rilasciando il codice sorgente della versione 5 di navigator in open source e cercando di creare una comunità online
  - 1998 – nasce mozilla.org e di conseguenza firefox
- la seconda guerra dei browser:
  - 2004 – microsoft si rifiuta di entrare nel gruppo WHAT, limitandosi a inseguire lo sviluppo delle tecnologie
  - 2008 – nasce chrome di google
  - 2015 – esce microsoft edge

    - 2020 – esce una versione di edge basata su chromium
- perdita di credibilità del W3C:
    - patent policy:
        - 2001 – sorge il problema delle imprese private che propongono la standardizzazione su tecnologie coperte da brevetti
        - spesso questa viene accettata dagli organismi di standard a patto che le tecnologie siano distribuite con una licenza RAND (reasonable and non discriminatory) che permette la competizione
        - 2004 – il W3C insiste per avere una policy RF (royalty free) salvo eccezioni (0)
        - alcune imprese ritirano la loro adesione al W3C, altre ritirano i membri dal working group, altre ancora continuano attivamente a partecipare, per velocizzare o rallentare le release dei working group a loro favore
    - il gruppo WHAT e lo sviluppo di HTML:
        - 2004 – il W3C boccia la proposta di raprire il working group su HTML per lo sviluppo di nuove versioni del linguaggio
        - si forma quindi il WHAT (web hypertext application technology), un gruppo separato, chiuso e finanziato dalle società di software
        - 2007 – il W3C riapre il working group con tutti i membri del WHAT per creare una nuova versione di HTML, HTML 5
- da HTML 5 a HTML:
    - HTML non è arrivato a una versione stabile sotto forma di W3C recommendation
- da controllare gli utenti a controllare gli sviluppatori:
    - le prime due guerre dei browser avevano lo scopo di ottenere siti web che non funzionavano sui browser della concorrenza
    - la nascita del W3C è nato infatti come ambiente di mediazione e soluzione delle dispute
    - è iniziata dopo la progettazione di browser come ambienti di esecuzione assolutamente generici
    - ora infatti la guerra si combatte sulle feautre scelte dallo sviluppatore, che gli permettono di sviluppare più facilmente e velocemente siti più sofisticati
    - in cambio lo sviluppatore accetta di bloccare in una specifica tecnologia il proprio sito (technology lock-in)
- caratteristiche del lock-in tecnologico:
    - tecnologie che:
        - forniscono servizi utili e graditi agli utenti
        - hanno un costo ragionevole o sono gratuite
        - hanno una forte personalità e cambiano il modo di realizzare i prodotti che supportano
        - cambiano il prodotto stesso che supportano
    - l'utente finisce per diventare dipendente dalla tecnologia
- la guerra attuale:
    - su librerie e framework per la creazione di applicazioni web
    - si utilizzano componenti per modularizzare e organizzare tutte le parti di un'applicazione web complessa
    - permettono di creare applicazioni web estremamente sofisticate e complesse con feature che avrebbero richiesto molto più lavoro
    - conducono inevitabilmente a un lock-in tecnologico

introduzione al www

- www:
    - sistema per la presentazione a schermo di documenti multimediali e per l'utilizzo di link

ipertestuali per la navigazione
    - distribuito e scalato su tutta internet
    - basato su:
        - client o browser: visualizzatore di documenti ipertestuali e multimediali
            - ha plug-in che permettono di visualizzare ogni tipo di formato specialee un linguaggio di programmazione inetrno (javascript) che permette di realizzare verifiche di dati o applicazioni autosufficienti (rich client)
        - server: meccanismo di accesso a risorse locali, in grado di trasmettere via socket TCP documenti individuati da un identificatore
            - può collegarsi ad applicazioni server-side e agire da tramite tra browser e applicazione (il browser diventa interfaccia dell'applicazione)
- protocolli fondamentali:
    - URI: standard per identificare in maniera generale risorse di rete e poterle specificare all'interno di documenti ipertestuali
    - HTTP: protocollo di comunicazione state-less e clilent-server per l'accesso a risorse ipertestuali via rete
    - HTML, ora XHTML: linguaggio per la realizzazione di documenti ipertestuali basato su SGML (ora XML)
- evoluzioni del www:
    - inclusione di oggetti
    - client scripting: per realizzare applicazioni client-side
    - stili tipografici: creazione di linguaggio appositi per gestire gli aspetti di visualizzazione del documento (CSS)
    - gestione delle transazioni: meccanismi per la gestione dello stato, ora standardizzati (cookies)
    - siti web dinamici: le applicazioni server-side diventano linguaggi o ambienti di prorammazione veri e propri
        - con l'arrivo delle librerie di accesso ai database nasce l'architettura a tre livelli (user-interface, application logic, data storage): il browser diventa interfaccia per applicazioni gestionali distribuite
    - strutturazione dei documenti: XML permette la definizione di linguaggi di markup più adatti ai singoli task
    - framework di sviluppo: ambienti integrati di sviluppo dotati di ricche API e librerie per la realizzazione semplificata di applicazioni client-side e server-side
- mode:
    - trucchi di html: per ottenere migliori effetti grafici
    - LAMP: risposta open source al modello DHTML/ASP di microsoft
        - stack di linguaggi e protocolli per la realizzazione di siti web dinamici e interattivi, con memorizzazione delle informazioni su un database relazionale, separazione di memorizzazione, logica e distribuzione
    - REST (representational state transfer): modello di interazione tra client e server proposto da HTTP
        - garantisce interoperabilità, scalabilità e uso delle caratteristiche di HTTP più avanzate
    - semantic web: attribuzione di semantica ai dati per la creazione e la condivisione di un modello concettuale delle informazioni
    - linked data: modello di attribuzione di semantica e indirizzabilità ai dati su cui si apossibile costruire applicazioni sofisticate e fortemente connesse anche in assenza di modelli concettuali forti
    - open linked data: versione politicizzata e politicamente corretta del linked data su dati

resi disponibili dalla pubblica amministrazione con modalità linked data
    - AJAX (asynchronous javascript and XML): meccanismo per generare applicazione web client-server fortemente interattive e in grado di minimizzare il traffico di rete
    - node.js: eseguibile che permette di eseguire codice javascript anche server-side
    - MEAN/MERN stack: nuovi stack di linguaggi e tecnologie per il web
    - mobile first: spostamento dell'attenzione verso i supporti mobili
    - responsive web design
    - single page websites
    - component-based design: parti visuali, markup, stile, computazione vengono sviluppate in maniera integrata per ogni componente e segregata tra componente e componente
        - un componente contiene frammenti di HTML (template), classi CSS e codice javascript, complessivamente autosufficienti e necessari per il funzionamente del componente
    - typescript, CLI, webpack, browserify:
        - typescript: linguaggio che transpila in javascript e aggiunge controlli sui tipi
        - CLI (command line interface): strumenti da command line, scriptabili e componibili, che rendono la configurazione di un progetto web veloce e aggiornabile automaticamente
        - webpack: web bundler, cioè un compilatore di applicazioni in tecnologie varie che genera un'applicazione web eseguibile
            - permette di compilare, riorganizzare o offuscare il codice, introdurre automaticamene librerie e dipendenze
        - browserify: strumento che permette di importare nel browser librerie
    - progressive web applications: applicazioni web sviluppate e caricate come normali pagine web, ma che si comportano come applicazioni native quando utilizzate su un dispositivo mobile
    - opinionated vs not-opinionated frameworks: i framework sono librerie che semplificano la progettazione di pagine e servizi web e uniformano le differenze dei singoli browser
    - static site generators: applicazioni che pre-eseguono script server-side e generano un sito web statico

URI (uniform resource identifier)

- principi architetturali del WWW:
    - identificazione: ogni elemento di interesse (risorsa) è identificato da un id globale (URI)
    - interazione: scambio di messaggi sulla rete
    - formato: disponibilità di applicazione e identificazione del formato della risorsa
- risorsa: qualunque struttura che sia oggetto di scambio tra applicazione all'interno di WWW
- WWW utilizza gli identificatori per:
    - immagini e oggetti inclusi nei documenti HTML
    - link ipertestuali in documenti ipertestuali
    - identificatori di namespace per documenti XML
    - identificatori di risorsa su cui esprimere affermazioni
    - identificatori di risorse di cui fornire firme crittografiche o valori hash
- URI:
    - accessibili tramite l'utilizzo di protocolli
    - sono per definizione:
        - URL (uniform resource locator): sintassi che contiene informazioni immediatamente utilizzabili per accedere alla risorsa
            - indirizzo della risorsa utilizzabile immediatamente per l'accesso

            - soggetto a modifiche non sostanziali del meccanismo di accesso
        - URN (uniform resource names): sintassi che permette una etichettatura permanente e non ripudiabile della risorsa
            - nome stabile e definitivo di una risorsa
            - deve essere trasformato da un apposito servizio nell'URL associato alla risorsa
    - la distinzione tra i due è secondaria: l'URN non si limita a uno schema possibile
- criteri di design degli URI:
    - trascrivibili: sono sequenze di caratteri presi da un set limitato
    - fornire identificazione, non interazione: la loro interpretazione fornisce le operazioni eseguibili sulle risorse e i protocolli per eseguirle
    - fornire spazi di nomi organizzati gerarchicamente: caratteri specifici separano diversi ambiti al loro interno
- organizzazione degli URI:
- URI = schema : [//authority] path [?query] [#fragment]
    - schema: stringa registrata presso IANA usata come prefisso (il protocollo negli URL)
    - authority: organizzazione gerarchica dello spazio dei nomi a cui sono delegati i nomi
        - authority = [userinfo @] host [: port]
            - userinfo: non deve essere presente se lo schema non prevede identificazione personale
            - host: nome di dominio o indirizzo IP
            - port: può essere omessa se ci si riferisce a una well-known port
    - path: parte identificativa della risorsa all'interno dello spazio di nomi, divisa in blocchi separati da /, ciascuno dei quali e un componente del path organizzato in gerarchia
    - query: specificazione della risorsa all'interno dello spazio di nomi
        - passa i parametri all'URI per specificare un risultato dinamico
        - forma nome1=valore1&nome2=valore+2+roba
    - fragment: indica una risorsa secondaria
- caratteri ammessi:
    - unreserved (non riservati): alfanumerici o di punteggiatura privi di ambiguità
    - reserved: che hanno funzioni particolari in uno o più schemi URI
        - vanno usati direttamente quando le assolvono, e da escaped quando fanno parte della stringa identificativa
    - escaped: %XX (maniera escaped), con XX codice esadecimale del carattere
        - caratteri non US_ASCII
        - caratteri di controllo
        - caratteri unwise
        - delimitatori
- route: associazione della parte path di un URI a una risorsa gestita o restituita da un server web
    - managed: il server associa ogni URI a una risorsa o attraverso il file system locale (risorse statiche) o attraverso computazione (risorse dinamiche)
    - file-system: il server associa la rdice della parte path a una directory del file system locale e ogni filename valido al suo intero genera un URI corretto e funzionante
- URI ref (reference):
    - fa riferimento a un URI di base rispetto al quale fornisce porzioni differenti
    - URI assoluto: contiene tutte le parti predefinite dal suo schema
    - URI gerarchico: può essere relativo, cioè riporta solo una parte dell'URI assoluto corrispondente
- operazioni su URI:

    - URI resolution: quando l'URI è una reference o uno a cui non risponde una risorsa fisica
        - è la generazione dell'URL assoluto corrispondente all'URI
    - URI deferencing: fornitura della risorsa identificata dall'URI
- risolvere una URI reference:
    - identificare l'URI assoluto cercato sulla base di URI relativo e URI di base
    - casistica:
        - inizia con #: è un frammento interno al documento di base
        - inizia con uno schema: è un URI assoluto
        - inizia con /: è un path assoluto all'interno dell'authority del documento di base
    - altrimenti:
        - si estrae il path assoluto dall'URI di base, meno l'ultimo elemento, e si aggiunge in fondo l'URI relativo
        - ./: si cancella
        - ../: si elimina insieme all'elemento precedente
- schemi HTTP e HTTPS:
    - unica differenza: HTTPS prevede una crittografia in entrambi i sensi del messaggio
    - sintassi:
        - http://host[:port]/path[?query][#fragment]
        - https://host[:port]/path[?query][#fragment]
            - host: indirizzo TCP-IP o DNS dell'host su cui si trova la risorsa
            - port: porta su cui il server è in ascolto (default: 80 HTTP, 443 HTTPS)
            - path: pathname gerarchico epr l'identificazione della risorsa
            - query: frase che identifica l'oggetto di ricerca sulla risorsa
            - fragment: identificativo di una sottoparte dell'oggeto, ignorata dal server perchè a carico del client
- schema FILE:
    - da accesso ai file di un file system locale
    - sintassi: file://host/path[#fragment]
    - assumendo che sia localhost, si può eliminare la parte host
        - sintassi: file:///path
- schema DATA:
    - schema non gerarchico che contiene la risorsa: tutti i dati sono inseriti nell'URI
    - usato solitamente per immagini inline su cui non si vuole attivare una connessione HTTP separata
    - sintassi: data:[<media type>][;base64;],<data>
        - media type: MIME registrato
        - base64: permette di codificare il dato
- FTP:
    - sintassi: ftp://[user[:password]@]host[:path]/path
        - user e password: necessari per l'accesso al server FTP, la mancanza di user porta a una connessione anonima
            - si scoraggia l'uso della password nell'URI per chiari motivi di sicurezza
        - host, port e path: indirizzo di server, porta di connessione (default 21) e nome del file dell'oggetto cercato
- URI che iniziano con //:
    - inizia con un nome dominio
    - va risolto in un URI assoluto utilizzando le parti mancanti dell'URI di base
    - usato solitamente nelle chiamate CDN
- CDN (content delivery network):

    - rete fortemente distribuita di server commerciali che collaborano tra loro per distribuire in maniera omogenea contenuti di grande successo senza duplicazioni di occupazione e trasmissione di file
    - l'uso permette di sfruttare al meglio le capacità di caching delle macchine end-user o dei nodi intermedi della rete
- pericolo delle illazioni a partire dalla forma degli URI:
    - le tecnologie attuali permettono di svincolare la forma dell'URI da tutti i dettagli connessi con la sua memorizzazione
    - ciò è fondamentale per parlare di risorse piuttosto che di file e processi
- URL permanenti:
    - PURL o permalink (esistono diversi schemi)
    - un sistema server-side prende un URL a cui non corrisponde una risorsa fisica e identific quale sia l'URL fisico corretto per la risorsa
    - accesso:
        - dereferenziazione: il sistema converte l'URI virtuale nell'URI fisico, accede alla risorsa e la restituisce
        - redirezione: il sistema fornisce al richiedente una risposta 302 temporary redirect e l'URI per accedere alla risorsa
- URI rewriting:
    - trasforma un URI visibile in un URL fisico
    - vantaggi:
        - permette di nascondere dettagli dell'implementazione
        - permette di realizzare sistemi di nomi perduranti oltre la vita del software
        - permette di esprimere informazioni semantiche sulla struttura del sito
- URI shortener:
    - compie una redirect
- application/x-www-form-urlencoded:
    - estensione della codifica degli URI applicata anche a risorse trasmesse su un canale HTTP
        - codici alfanumerici sostituiti da %HH (HH codice esadecimale del carattere)
        - spazi sostituiti da +
        - nomi dei controlli separati da &
        - valore separato dal nome con =
    - usato dai form che trasmettono dati a server
- IRI (internationalized resource identifier):
    - fornisce una sintassi per inserire caratteri di UCS-4 in un URI e risolverli
        - altrimenti erano disponibili solo caratteri ASCII 7 bit, o con escaping ISO latin-1 (non per forza visibili a schermo)
    - non sono ammessi i codici di controllo e quelli non corrispondenti a caratteri usati nelle lingue
- IDNA (internazionalized domain name for application): standard IETF per estendere i nomi di dominio a tutti i caratteri non ASCII di unicode
    - IDN: risultanti nomi di dominio
    - rischi: spoofing risultato dell'esistenza di caratteri simili in script diversi
        - alcuni registri di nomi non accettano domini con caratteri di script diversi
- CURIE (compact URI):
    - modo di esprimere in maniera compatta famiglie di URI con lo stesso prefisso
        - per evitare spreco di spazio e rischio di errori di compilazione dovuti a una lunghezza considerevole degli URI e alla possibile presenza di URI simili in parti

diverse della stessa risorsa
    - sintassi: [prefix:curie]
    - il prefisso è associato a un URI assoluto che deve essere sostituito nel CURIE per ottenre l'URI cercato
- linked data:
    - modello concettuale per la caratterizzazione di tutte le risorse di dati che possono essere messi a disposizione e incrociati da applicazioni e umani
        - necessario per la grande quantità di informazioni strutturate presenti nella rete
    - una collezione di dati deve:
        - usare URI per identificare oggetti
        - usare HTTP URI in modo che questi oggetti possano essere referenziati e cercati da persone e user agent
        - fornire informazioni utili sull'oggetto quando la sua URI è dereferenziata
        - includere link ad altri URI relativi ai dati esposti per migliorare la ricerca di altre informazioni relative
- linked open data:
    - linked data rilasciato con licenza aperta

character encoding
- digitalizzazione di dati non numerici:
    - associazioni di un numero identificativo a un dato
        - numero univoco per ogni carattere diverso
    - si utilizza la tecnica divide et impera
- set di caratteri:
    - bisogna rendere in modo corretto tutti gli alfabeti esistenti (e non) del mondo
    - il problema si pone nel contenuto dei protocolli: il criterio di associazione di un blocco di bit a un carattere deve essere evidente e non ambiguo
- rappresentazione binaria del testo:
    - rappresentazione digitale della scrittura, ottenuta con:
        - identificazione degli elementi fondanti (caratteri)
        - identificazione dello spazio di rappresentazione
        - creazione di un mapping e ufficializzazione
- caratteri:
    - entità atomiche di un testo scritto
    - in diversi alfabeti hanno particolarità diverse
    - in diverse lingue hanno ruoli diversi: possono rappresentare suoni, sillabe o parole
    - aspetti:
        - natura
        - forma (o glifo)
        - codice numerico
- regole più importanti della digitalizzazione:
    - ordine: i valori seguono l'ordine alfabetico
    - contiguità: ogni valore numerico compreso tra il più basso e il più alto è associato a un carattere
    - raggruppamento: l'appartenenza a un gruppo logico è riconoscibile da considerazioni numeriche
- termini:
    - shift: codice riservato che cambia mappa dal momento in poi
    - codici liberi: non associati, la lor presenza in un flusso di dati indica probabilmente un

errore di trasmissione
    - codici di controllo: associati alla trasmissione
- baudot:
    - codice a 5 bit (32 codici possibili), ma attraverso l'uso di un codice per lo shift di lettere e uno per lo shift di numeri, si arriva a 64 codici possibili:
        - 50 per lettere, numeri e punteggiatura
        - 9 codici di controllo
        - 2 shift
        - 3 codici liberi
    - non è contigua nè ordinata
- ASCII (american standard code for information interchange):
    - 128 caratteri (7 bit su 8, nell'originale, il primo bit viene usato per la parità)
        - 95 caratteri dell'alfabeto latino
        - 33 caratteri di controllo (tra cui ripetizioni)
        - 0 codici liberi
    - codifica contigua e ordinata
- EBCDIC (extended binary characters for digital interchange code):
    - codifica proprietaria (IBM) a 8 bit
    - 56 codici di controllo e molte locazioni vuote
    - lettere dell'alfabeto non contigue ma organizzate in modo da avere il secondo semibyte da 0 a 9
- ISO 646-1991:
    - permette l'uso di caratteri nazionali europeri in un contesto ASCII
    - ha una IRV (international reference version) identica all'ASCII e un tot di versioni nazionali
    - 12 codici liberi per le versioni nazionali dei linguaggi europei
- le code page di ASCII:
    - meccanismo di estensioni multiple e indipendenti di ASCII per le necessità di script, alfabeti e usi diversi
    - estensioni di ASCII per usare i rimanenti 128 caratteri (0-127 sono di ASCII)
- CP 737: per l'alfabeto greco
- KOI7 e KOI8: per il cirillico
    - la versione a 7 bit usa due caratteri switch per passare ai caratteri latini
- codifiche CJK (cinese, giapponese, coreano): codifiche a 16 bit o più larghe ancora
    - regge almeno i caratteri han più script fonetici specifici
- ISO latin 1: unica estensione standard per accedere a tutti i 256 caratteri
    - usato di default da HTTP e alcuni SO
    - compatibile all'indietro con ASCII
- esigenza di uno standard nazionale:
    - essendoci molte codifiche a 8 bit (alfabet latini e non) e a 16 bit (linguaggi orientali), c'è la necessità di meccanismi indipendenti dal flusso per specificare il tipo di codifica usata
    - nei flussi misti si adottano meccanismi di shift da una codifica all'altra, dipendenti dall'applicazione
- unicode e ISO/IEC 10646:
    - commissioni per la creazione di uno standard unico
    - definiscono codifiche a lunghezza fissa o variabile
    - le differenze tra i due sono scomparse
    - ultima versione:
        - definisce quasi 140.000 caratteri diversi + 140.000 per scopi privati (PUA, private

use areas)
        - categorie:
            - script moderni
            - script antichi
            - segni speciali
- conscritpt unicode registry (scripts for construted languages): catalogo ufficioso di assegnazioni
- principi di unicode:
    - repertorio universale
    - efficienza: minimo uso di memoria per massima velocità di parsing
    - caratteri, non glifi: font esclusi
    - semantica: ogni carattere possiede un proprio significato e proprietà
    - testo semplice: codepoint (caratteri di testo semplice, senza descrizioni grafiche o tipografiche)
    - ordine logico: le sottosequenze seguono l'ordine naturale alfabetico
    - unificazione: caratteri comuni a linguaggi diversi, se possibile, vengono unificati
    - composizione dinamica:
        - i caratteri che sono composizioni di frammenti indipendenti, hanno codici indipendenti e vengono creati per composizione
        - i più comuni hanno una sequenza equivalente (codice singolo equivalente)
    - stabilità: i codici rimangono immutabili e non si possono rimuovere
    - convertibilità: c'è un meccanismo di conversione tra unicode e codifiche precedenti
- ISO/IEC 10646:
    - due schemi di codifica:
        - UCS-2: a 2 byte
        - UCS-4: a 31 bit in 4 byte, diviso in gruppi, piani, righe e celle
            - esistono 32768 piani di 65536 caratteri ciacuno
                - piano 0 (BMP, basic multilingual plane): equivalente a UCS-2, alfabeti moderni
                - piano 1 (SMP, supplementary multilingual plane): alfabeti antichi
                - piano 2 (SIP, supplementary ideographic plane): caratteri dieografici CJK non presenti in BMP
                - piano 3 (TIP, tertiary ideographic plane): caratteri cinesi antichi
                - piano 14 (SSP, supplementary special-purpose plane): caratteri tag (in disuso)
                - piani 15 e 16: PUA
- UTF (unicode transformation format o UCS transformation format): sistema a lunghezza variabile che permette di accedere a tutti i caratteri di UCS in maniera più semplice e efficiente
- UTF-8: permette di accedere a tutti i caratteri definiti in UCS-4, ma utilizza un numero 1-4 di byte per farlo
    - i codici 0-127 (ASCII) richiedono un byte con 0 al primo bit
    - i codici di alfabeto latino e script non ideografici richiedono 2 byte con i primi due bit a 11
    - i codici ideografici richiedono 3 byte coi primi tre bit a 111
    - i codici dei piani alti (in UTF-16 utilizza coppie di surrogati, cioè caratteri appartenenti a un piano non BMP) richiedono 4 byte coi primi quattro bit a 1111
    - il secondo byte e gli altri contengono la sequenza 10 (continuation bit) e 6 bt significativi
    - se un byte inizia per 10, è di continuazione a va ignorato

- little-endian, big-endian:
  - alcuni processori generano e gestiscono i flussi di coppie di byte ponendo il byte più significativo prima, altri dopo il byte meno significativo
  - causa di effetti sulle capacità di interpretare correttaente flussi di byte provenienti da processori ignoti
- BOM (byte order mark):
  - unicode specifica un segnalatore di ordinamento del flusso: FFFE (little-endian) o FEFF (big-endian)
  - FEFF è il carattere zero-width no-break space (ZWNBSP): usato come whitespace, non modifica il significato dei testi
    - in little-endian è proibito in unicode
      - quindi lo si utilizza all'inizio di ogni flusso UTF-16 e UCS-2, in modo da identificare se si big-endian o little-endian
- differenze tra UTF-8 e ISO latin-1:
  - per i caratteri di ASCII non ci sono differenze
  - UTF-8 utilizza 2 byte per lettere latine con decorazioni, mentre ISO latin-1 1

HTTP (hypertext transfer protocol)
- protocollo client-server, generico e stateless:
  - client-server: il client attiva la connessione e richiede i servizi, il server accetta la connessione (identificando il richiedente nel caso), risponde alla richiesta e chiude la connessione
  - generico: è indipendente dal formato dati con cui vengono trasmesse le risorse
  - stateless: il server non è tenuto a mantenere informazioni che persistano tra una connessione e l'altra; il client è tenuto a ricreare da zero il contesto necessario al server per rispondere
- risorse HTTP:
  - permette lo scambio di risorse identificate da URI
  - separa le risorse dalla loro rappresentazione
  - fornisce meccanismi di negoziazione del formato di dati
  - implementa politiche di caching che permettono di memorizzare copie delle risorse sui server coinvolti e controllarne la validità
- ruoli delle applicazioni HTTP:
  - client: stabilisce una connessione HTTP con lo scopo di mandare richieste
  - server: accetta connessioni HTTP e genera risposte
  - user agent: client che inizia una richiesta HTTP
  - origin server: server che possiede fisicamente la risorsa richiesta
  - proxy: applicazione intermediaria che agisce sia da client che da server; soddisfa le richieste autonomamente o passandole ad altri server
  - gateway: applicazione che agisce da intermediario per altri server; riceve le richieste come fosse l'origin server
  - tunnel: programma intermediario che agisce da trasmettitore passivo di una richiesta HTTP
- connessione HTTP:
  - persistente
  - pipelining: trasmissione di più richieste senza l'ack delle richieste precedenti, le risposte sono restituite in ordine di invio
  - multiplexing nella stessa connessione: richieste e risposte multiple restituite anche in ordine diverso
    - più operazioni in parallelo quindi meno latenza

- HTTP/2:
  - obiettivo: ridurre i tempi di latenza di HTTP
  - introduce:
    - multiplexing
    - compressione messaggi (non più plaintext) e separazione del blocco degli header dal payload
    - supporto delle operazioni di push da parte del server, che può spedire più dati di quelli richiesti anticipando richieste successive sulla stessa connessione
- HTTP/3:
  - obiettivo: aumentare le performance rispetto a HTTP/2
  - costruito su UDP (user datagram protocol)
- metodi (o verbi) HTTP:
  - azione che il client richiede al server sulla copia della rappresentazione della risorsa
  - permettono la creazione di applicazioni interoperabili e in grado di sfruttare al meglio i meccanismi di caching
  - proprietà:
    - sicurezza: se non genera cambiamenti allo stato interno del server (a parte i log)
      - può essere eseguito da un nodo inetrmedio senza effetti negativi
    - idempotenza: se l'effetto sul server di più richieste identiche è lo stesso di quello di una sola richiesta (a parte i log)
      - può essere ri-eseguito da più agenti o in più tempi diversi senza effetti negativi
  - GET: richiesta di una risorsa a un server
    - sicuro e idempotente
  - HEAD: simile a GET, ma il server risponde solo con header relativi senza il corpo
    - usato per verificare validità, accessibilità e coerenza in cache di un URI
    - sicuro e idempotente
  - POST: trasmette informazioni da client a server relative alla risorsa identificata nell'URI
    - può anche essere usato per creare nuove risorse
    - non è sicuro nè idempotente
  - PUT: trasmette informazioni da client a server creando o sostituendo la risrsa specificata
    - l'argomento è la risorsa che ci si aspetta di ottenere facendo un GET con lo stesso nome in seguito
    - non offre garanzie di controllo di accessi o locking
    - idempotente ma non sicuro
  - DELETE: rimuove le informazioni connesse a una risorsa
    - un GET in seguito sulla stessa risorsa risulterà in un error 404
    - un DELETE su una risorsa non esistente è lecito e non genera errore
    - idempotente e non sicuro
  - PATCH: aggiorna parzialmente la risorsa identificata dall'URI
    - indica modifiche incrementali da effettuare sulla risorsa
    - non è sicuro nè idempotente
  - OPTIONS: verifica opzioni, requisiti e servizi di un server, senza implicare una richiesta successiva
    - usato per il problema del cross-site scripting
    - permettere richieste a domini diversi da quello su cui risiede la pagina
    - verifica se una eventuale richiesta futura verrà accettata sulla base dell'identità del richiedente (a differenza di HEAD, che lo fa in base alla risorsa richiesta)
    - sicuro e idempotente
- richiesta HTTP:

    - method
    - URI
    - version
    - header:
        - generali: informazioni sulla trasmissione
            - si applicano al messaggio trasmesso ma non necessariamente alla risorsa
        - di entità: informazioni su risorsa e dati trasmessi
            - danno informazioni sul body del messaggio o (se non c'è body) sulla risorsa specificata
        - di richiesta: informazioni sulla richiesta effettuata
            - messo dal client per specificare informazioni su richiesta e se stesso al server
            - usato per gestire cache e meccanismi di autenticazione
    - body: messaggio MIME
- riposta HTTP:
- status code: indica se la richiesta è andata a buon fine
    - numero di tre cifre, di cui la prima indica la classe della risposta e le altre due la risposta specifica
    - classi:
        - 1xx – informational: risposta temporanea durante lo svolgimento della richiesta
        - 2xx – successful: il server ha ricevuto e accettato la richiesta
        - 3xx – redirection: la richiesta è corretta, ma sono necessarie altre azioni da parte del client per portarla a termine
        - 4xx – client error: la richiesta non può essere portata a termine per un errore (sintattico o richiesta non autorizzata) del client
        - 5xx – server error: la richiesta può essere corretta, ma il server non la soddisfa per problemi interni
    - utilità:
        - aiuta a costruire API chiare e semplici da usare
        - diminuisce la dipendenza dal server: il client non ha bisogno di leggere il body della risposta ma basta leggere lo status code
        - permette a tutte le entità nella comunicazione di capire cosa succede e sfruttare al meglio i meccanismi di caching e redirezione HTTP
        - migliora uniformità e interoperabilità
- version
- header:
    - generali
    - di entità
    - di risposta: informazioni sulla risposta generata
        - messo dal server per specificare informazioni su risposta e se stesso al client
        - alcuni definiscono il tipo di dato contenuto nella risposta, permettendo al client di processare correttamente la risorsa
        - se viene fornita un'entità in risposta, sono obbligatori:
            - content-type: usato dallo user agent per sapere come visualizzare l'oggetto ricevuto
            - content-length: usato dallo user agent per sapere di aver ricevuto l'intero oggetto
- body: messaggio MIME

API (interfaccia di programmazione applicazioni)

- insieme di metodi e proprietà pubblici che utilizza un oggetto per interagire con altri oggetti

nell'applicazione
    - URL dedicato che restituisce risposte in forma di puri dati (senza presentazione grafica)
- applicazione: può riferirsi a qualsiasi parte di software sia distintamente separata dall'interfaccia
- YAML (YAML ain't a markup language):
    - linearizzazione di strutture dati simile a JSON ma con sintassi ispirata a python:
        - superset di JSON
        - indentazione
        - supporto di tipi scalari, liste e hash
        - supporto di stringhe multiriga con due delimitatori diversi
        - supporto di commenti

REST (representational state transfer)
- modello architetturale che sta dietro al www
    - modo di strutturare le applicazioni per sfruttare pienamente le caratteristiche di HTTP
- applicazioni non REST:
    - basate su:
        - generazione di API che specifichi le funzioni a disposizione dell'applicazione
        - creazione di un'interfaccia indipendente dal protocollo di trasporto e ad essa nascosta
- applicazione REST:
    - basata su: uso di protocolli di trasporto e di naming per generare interfacce generiche di interazioe con lìapplicazioni e connesse con l'ambiente d'uso
- API web: descrive un'interfaccia HTTP che permette ad applicazioni remote di utilizzarne i servizi
- modello CRUD: pattern tipico delle applicazioni di trattamento dei dati
    - ipotizza che tutte le operazioni dei dati siano di tipo:
        - create: inserimento nel database di un record nuovo
        - read: accesso in lettura
            - individuale o contenitore (gruppo di individuali in base a una proprietà)
        - update
        - delete
- architettura REST:
    - basata su:
        - definire risorsa ogni concetto rilevante dell'applicazione web
        - associargli un URI come identificatore e selettore primario
        - usare i verbi HTTP per esprimere ogni operazione dell'applicazione secondo il modello CRUD:
            - creazione -> PUT, visualizzazione -> GET, cambio -> POST, cancellazione -> DELETE
        - esprimere in maniera parametrica ogni rappresentazione dello stato interno della risorsa, personalizzabile dal richiedente attraverso un content type preciso
- approccio web service tradizionale:
    - modello SOAP, per il quale esiste un intermediario di messaggi
        - a questo viene indirizzata la richiesta che contiene nel body tutte le informazioni necessarie a soddisfarla
    - HTTP viene usato solo ed esclusivamente per il trasferimento trasparente di informazioni
- individui e collezioni:
    - concetti fondamentali ai quali viene fornito un URI e ogni operazione avviene su uno

solo di questi
    - su entrambi si possono eseguire operazioni CRUD
- gerarchie: rendono le API più leggibili e il routing semplificato in molto framework di sviluppo
- URI in REST:
    - le collezioni sono intrinsecamente plurali, mentre gli individui intrinsecamente singolari
    - filtro: genera un sottoinsieme specificato attraverso una regola (query)
    - uso dei verbi HTTP:
        - GET: elenco, accesso a dati
        - PUT: creazione di un record (il client decide l'id) o modifica di (tutti i) dati
        - POST: creazione di un record (id non deciso dal client) o modifica di alcuni dati
            - può essere usato in varie situazioni secondo una semantica decisa localmente, purchè non sovrapposta ad altri verbi
        - DELETE: cancellazione di un record
- linee guida:
    - adottare una convenzione di denominazione chiara
    - usare gerarchie valutando i livelli necessari
    - evitare la creazione di API che rispecchiano la struttura interna del database
    - fornire meccanismi per filtrare e paginare le risposte
    - supportare richieste asincrone restituendo il codice HTTP 202 (accettato ma non completato) e le informazioni per accedere allo stato della risorsa
- controllo delle versioni di un API:
    - dato che REST non ha linee guida stringenti, esistono due approcci principali:
        - memorizzare il numero di versione all'inizio dell'URI
            - viola l'idea di identificare solo una risorsa nell'URI
        - usare header HTTP accept e i meccanismi di content negotiation per specificare la versione supportata
            - gli URI sono più puliti ma la complessità sul client aumenta
- HATEOAS (hypermedia as the engine of application state):
    - principio che guida il modello di interazione tra server e client
    - applicazione del modello ipertestuale anche allìidentificazione delle operazioni possibili su una rappresentazione di risorsa
- descrizione di RESTful API:
    - una API è RESTful se:
        - descrive gli end-point (URI/route) che supporta, separando colllezioni e individui
        - descrive i metodi HTTP di accesso
        - descrive le rappresentazioni in I/O
        - descrive tutte le condizioni di errore e i messaggi che restituisce in questi casi
    - modelli di maturità di API:
        - è normale trovare API ch dichiarano di essere RESTful ma non ne rispettano completamente le linee guida
        - si utilizzano quindi modelli di maturità ottenere livelli di rispetto
        - di richardson:
            - livello 0: POX
            - livello 1: URI
            - livello 2: HTTP
            - livello 3: hypermedia
- documentazione di un'API:
    - meccanismo più affidabile e manutenibile per creare un contratto e un collegamento

affidabile tra servizi offerti da un server e applicazioni realizzate da un client
        - soprattutto in caso di decoupling tra client e server
    - OpenAPI: standard di documentazione
        - nasce per:
            - facilitare la manutenzione di documentazione, test e implementazione di un'API
            - rendere le API human-readable
            - costruire API che siano indipendenti da specifici linguaggi di programmazione
        - specifica:
            - per creare servizi web RESTful
            - standard industriale per API REST
            - le applicazioni che lo usano possono generare automaticamente documentazione di metodi, parametri e modelli
        - operazioni:
            - tutti i percorsi sono relativi all'URL del server API:
            - l'URL della richiesta è <server-url>/path
            - API input:
                - almeno cinque parametri:
                    - path: parametri di percorso /users/{id}
                    - query: parametri di query /items?offset=num&limit=num
                        - tipicamente usato con get
                    - header: parametri di intestazione X-request-ID
                        - tipicamente usato con get
                    - cookie: parametri di cookie, passati nell'header di un cookie
                    - body: permette di passare come input interi oggetti
                        - tipicamente usato con post, put, delete
                - definiti attraverso le keyword parameters
                - per ognuno si può definire tipo (keyword in), nome (name), se sono opzionali (required), descrizione (description) e che caratteristiche/struttura e tipi di dato deve avere l'input (schema)
                - schema: può essere definito anche all'esterno dell'API, in modo da poterlo usare in più API con il suo riferiment $ref:"/definitions/nome_oggetto"
            - API output:
                - definiti con keyword responses
                - ogni risposta ha un id numerico univoco:
                    - 200 se non ci sono stati errori, dal 400 in su se ci sono stati errori
    - swagger: ecosistema di tool per creazione, costruzione, documentazione e accesso ad API soprattutto in ambito REST
    - GraphQL: alternativa globale alle API in cui il client chiede più dati e il risultato si ottiene da una negoziazione col server
        - nei modelli REST tradizionali è il server che decide quali URI accettare e quali rappresentazioni associare a questi
            - ciò introduce rigidità ed eccesso di informazioni trasmesse
        - il client elenca attraverso un esempio il modello di dati che sta cercando, e il server restituisce i dati nel formato JSON richiesto

content encoding

- certi ambienti informatici forniscono restrizioni sulla varietà di caratteri usabili:
    - modelli di rappresentazioni dei dati: alcuni caratteri hanno scopi tecnici e non possono essere usati come contenuto

    - canali di trasmissione: non sono trasparenti all'uso di flusso di dati a 8 bit (8 bit clean)
- termini:
    - escaping: il carattere proibito viene preceduto o sostituito da una sequenza di caratteri speciali
    - encoding: il carattere proibito viene rappresentato numericamente col suo codice naturale secondo una sintassi speciale
- l'origine dei problemi: SMTP (simple mail transfer protocol):
    - limiti: impediscono la trasmissione di documenti binari, tra gli altri
        - lunghezza massima del messaggio a 1 Mb
        - caratteri accetta solo da ASCII a 7 bit
        - ogni messaggio deve contenere una sequenza CRLF ogni 1000 caratteri o meno
- MIME (multipurpose internet mail extensions):
    - permette di bypassare i limiti di SMTP
    - ridefinisce il formato del corpo in modo da permettere:
        - messaggi di testo in altri set di caratteri al posto di US-ASCII
        - insieme estensibile di formati per messaggi non testuali
        - messaggi multi-parte
        - header con set di caratteri diversi da US-ASCII
- messaggi MIME su canali SMTP:
    - il messaggio non compatibile viene trasformato in uno o più messaggi SMTP da un preprocessore al server SMTP
    - il messaggio viaggia sul canale puro SMTP
    - all'arrivo il messaggio viene decodificato e, nel caso, riaccorpato
- limiti SMTP su MIME:
    - codifica caratteri: un messaggio non ASCII a 7 bit viene codificato come serve, il transfer encoding è diverso per messaggi di testo o binari
    - sequenze CRLF: tutti i sistemi di transfer encoding adottano un meccanismo per permettere la presenza di sequenze CRLF in mezzo al flusso dati
    - lunghezza messaggi: un processore MIME può generare vari messaggi SMTP da uno singolo, ciascuno inferiore pe dimensione al limite SMTP
        - il processore all'arrivo ha il compito di verificare l'arrivo corretto di tutti i singoli messaggi SMTP e ricorstruire il MIME originario
- servizi MIME:
    - dichiarazione di tipo: tutti i messaggi MIME vengono identificati da un content type
    - messaggi multi-tipo: un messaggio MIME può contenere parti di tipo diverso
        - si creano dei sottomessaggi per ciascuna e ogni parte viene codificata in maniera appropriata
- header specifici MIME:
    - nuovi header SMTP:
        - content-type: tipo MIME del contenuto
        - content-transfer-encoding: tipo di codifica utilizzata per trasmettere i dati
- tipi di transfer encoding definiti da MIME:
    - quoted printable: usato per la trasmissione di dati che contengono grandi quantità di byte nel set US-ASCII
        - codifica solo i byte non conformi
    - base 64: suggerito per dati binari o multi-byte
        - identifica un sottoinsieme di 64 caratteri di US-ASCII sicuri: lettere, numeri, + e /
        - ogni flusso di dati viene suddiviso in blocchi di 24 bit, suddivisi in 4 blocchi da 6 bit e codificati secondo una tabella prefissata in uno dei 64 caratteri

        - la stringa risultato viene divisa in righe di 76 caratteri + CR-LF
        - nella decodifica CR e LF vengono ignorati

markup

- ogni mezzo per rendere esplicita una particolare interpretazione di un testo
    - per sistemi informatici: specifica le modalità esatte di utilizzo del testo nel sistema stesso
- comprende delimitatori di parola, di frase e di periodo, numerazione di pagine e uso dei margini
- piramide dei linguaggi (conversione facilitata a scendere):
    - XML; DTP; TeX, LaTeX; word processor; ASCII; bitmap; carta
- modi:
    - proprietario vs pubblico:
        - proprietario: creato da un'azienda (che ne detiene i diritti) con uno specifico scopo commerciale
        - pubblico: creato da un gruppo di interesse come modello di armonizzazione tra le esigenze di ogni partecipante
            - in genere si pubblicano le specifiche del formato, permettendo a chiunque di utilizzarlo
        - a volte diventa uno standard ufficiale
    - binario vs leggibile:
        - binario: memorizzazione esatta delle strutture in memoria dell'applicazione, il testo non è visibile se non in casi specifici
        - leggibile: fatto per essere leggibile anche da esseri umani (per interventi di emergenza)
            - l'applicazione deve quindi trasformare quello che legge in una struttura interna utile per le operazioni di modifica o presentazione (= parsing)
    - interno vs esterno:
        - interno: inserisce istruzioni di presentazione all'interno del testo, in mezzo alle parole
            - richiede sintassi particolari per distinguere il markup dal contenuto
                - in genere, segnalatori che cambiano il tipo di interpretazione del documento
                    - la loro presenza richiede l'adozione di tecniche di escaping
        - esterno: prevede due blocchi di informazioni separati e collegati da indirezione (contenuto e markup)
            - richiede quindi un meccanismo di indirezione basato su indirizzi, offset e identificatori
                - per associare con correttezza il markup al contenuto
    - assolve diversi ruoli a seconda del sistema di elaborzione, dell'applicazione, dello scopo del documento:
        - puntuazionale:
            - consiste nell'usare un insieme prefissato di segno per fornire informazioni soprattutto sintattiche sul testo
            - regole di punteggiatura stabili, note e frequenti nei documenti
            - problemi nell'uso della punteggiatura:
                - incertezze strutturali e grafiche
                - ambiguità procedurali
        - presentazionale:
            - consiste nell'indicare effetti (grafici o altro) per rendere più chiara la presentazione del contenuto
        - procedurale:

        - consiste nell'indicare con precisione a un sistema automatico che effetti attivare e che procedure (serie di istruzioni) eseguire nella visualizzazione del contenuto
        - utilizzo quindi le capacità del sistema di presentazione per avere l'effetto voluto
    - descrittivo:
        - consiste nell'identificare strutturalmente il tipo di ogni elemento del contenuto
        - non specifico effetti grafici, ma ne individuo il ruolo, giustificazione, relazione con altri elementi, nel documento
    - referenziale: consiste nel fare riferimento a entità esterne al documento per dare significato o effetto grafico a elementi del documento
    - metamarkup: consiste nel fornire regole di interpretazione del markup e permette di estendere o controllarne il significato
  - senza markup -> metabolizzato (aggiunta di puntuazionale e presentazionale) -> procedurale -> descrittivo
- linguaggi di markup:
- TROFF/NROFF (GROFF su linux):
  - usato per documentazione tecnica
  - esiste un formatter in grado di creare documenti stampabili su stampante (troff) o schermo a carattere (nroff)
  - comandi:
    - esterni: compaiono su righe autonome precedute da un punto
    - interni: introdotti da \ (carattere di escape)
  - permette la creazione di macro complesse
- Tex e LaTeX:
  - TeX: linguaggio di programmazione completo, dotato di comandi per la formattazione di testi (300 comandi, primitive)
    - permette la generazione di macro che semplifican la generazione di testi sofisticati
    - ha librerie per la scrittura di documenti complessi specifici
    - metafont: sistema associato a TeX per la descrizione delle forme dei caratteri di un font attraverso formule matematiche
  - LaTeX: raccolta di macro TeX per la generazione di una ventina di tipi di documenti comuni
- SGML (standard generalized markup language):
  - metalinguaggio standard (quindi non proprietario) di markup descrittivo
  - facilita markup leggibili, generici, strutturati, gerarchici, descrittivi
    - leggibile: markup a fianco degli elementi del testo a cui si riferisce
    - gerarchico: le strutture (possibili grazie al markup strutturato) sono in genere a livelli di dettaglio successivi
      - gli elementi possono comporsi gli uni con gli altri permettendo di specificare la struttura in modo gerarchico
  - documenti SGML:
  - composti da tre parti:
  - dichiarazione SGML:
    - contiene le istruzioni di partenza delle applicazioni SGML
    - specifica i valori fondamentali per lunghezze e sintassi
    - è lunga varie centinaia di righe
    - non è obbligatoria: se è assente, si usa reference concrete syntax (una dichiarazione di default)
    - <!SGML ...>
  - dichiarazione di documento (document type declaration, DTD):

        - specifica le regole che permettono di verificare la correttezza strutturale di un documento
        - elenca gli elementi ammissibili, il contesto in cui possono apparire e altri vincoli strutturali
            - modella quindi una classe di documenti attribuendogli un tipo
        - <!DOCTYPE nome [regole]>
    - istanza del documento:
        - testo vero e proprio col markup appropriato
        - le applicazioni SGML sono in grado di verificare se l'istanza segue le regole specificate nel DTD e di identificarne le violazioni
- metalinguaggio di markup:
    - linguaggio per definire linguaggi, grammatica di costruzione di linguaggi
    - fornisce una sintassi per definire il linguaggio adatto
- componenti del markup:
    - elementi: parti di documento dotate di un senso proprio
        - ognuno è individuato da un tag iniziale, un contenuto e un tag finale
    - attributi: informazioni aggiuntive sull'elemento (meta-informazioni)
        - posti dentro al tag iniziale dell'elemento (in genere nome=valore)
    - entità: frammenti di documento memorizzati separatamente e richiamabili all'interno del documento
        - permettono di riutilizzare lo stesso frammento in diverse posizioni garantendo l'estatta corrispondenza dei dati e permettendo una modifica semplificata
    - testo: contenuto del documento, include parole, spazi e punteggiatura
        - anche detto #PCDATA (parsed character data)
            - i linguaggi di markup definiscono character data (CDATA) il contesto testuale, e quello degli elementi è soggetto a parsing
    - commenti: note che le applicazioni d markup ignorano
    - processing instructions (PI): elementi messi da autore o applicazione per dare indicazioni su come gestire il documento nel caso specifico
- XML 1.0:
    - raccomandazione W3C definita come sottoinsieme di SGML
    - più formalizzata della grammatica di SGML
    - usa la extended backus-naur form (notazione formale)
- XML distingue:
    - documenti ben formati:
        - pur privi di DTD, hanno una struttura abbastanza regolare e comprensibile da poter essere controllata
        - se:
            - tutti i tag di apertura e chiusura corrispondono e sono ben annidati
            - esiste un elemento radice che contiene tutti gli altri
            - gli elementi vuoti utilizzano un simbolo speciale di fine tag
            - tutti gli attributi sono chiusi tra virgolette
            - tutte le entità sono definite
    - documenti validi:
        - se presenta un DTD ed è validabile con questo
        - in SGML è infatti necessario il DTD per la validazione
- markdown e sintassi wiki: formati testuali che usano trucchitestuali ad hoc per ottenere strutture ed effetti tipografici precisi
- JSON (javascript object notation): formato dati derivato dalla notazione usata da JS per gli

oggetti

HTML

- due linee parallele di HTML:
  - una recommendation del W3C
  - HTML living standard del WHATWG
- HTML 4:
  - tipo di documenti SGML progettato per marcare documenti ipertestuali
  - rende possibile realizzare documenti con una semplice struttura, con aspetti grafici sofisticati, con oggetti interattivi e link
  - aggiunge il supporto per l'internazionalizzazione, per gi style sheet, per i frame e tabelle più ricche
- tag soup: insieme di elementi non conformi allo standard
  - quindi esistono differenze tra un documento corretto e uno visualizzabile da un browser web
  - non è mai stato standardizzato (prima di HTML 5) il modo in cui fare il parsing di questi documenti
- per gestire sia le pagine conformi all standard sia le altre, sono stati introdotti due modelli di rendering:
  - quirks mode: compatibile col passato e più permissivo
    - adottato se non c'è la specifica di strict mode
  - strict mode: compatibile con le specifiche ufficiali
- XHTML 1.0:
  - riformulazione di HTML 4 come un'applicazione di XML 1.0
  - elenco e semantica di elementi e attributi uguale a HTML 4
  - differenze da HTML 4:
    - nomi di elementi e attributi minuscoli
    - tag finale obbligatorio per elementi non vuoti
    - gli elementi vuoti devono seguire la sintassi XML
    - i valori degli attributi devono avere le virgolette
- HTML 5:
  - specifica in perenne sviluppo (living standard)
  - cambia quindi il modello di sviluppo del linguaggio rispetto agli altr standard del W3C
  - novità:
    - il linguaggio di markup è stato rivisto globalmente, introducendo nuovi elementi più specifici per usi consolidati
    - uso di script per la modifica dinamica in memory del documento (DOM)
    - HTML definisce e descrive un certo numero di API per una caratterizzazione del DOM più sofisticata rispetto a quella col markup
    - XHTML 5: linearizzazione facoltativa di HTML basata sulla sintassi XML
-