Tecnologie Web

Introduzione al World Wide Web

Il W3C (World Wide Web Consortium) nasce dall'iniziativa di Tim Berners-Lee nel 1994 per promuovere e uniformare lo sviluppo delle tecnologie del Web.

- La patent policy: nel 2001 sorse il problema delle imprese private che proponevano la standardizzazione su tecnologie coperte da brevetti. Il W3C, invece, preoccupato di perdere il supporto degli sviluppatori open-source, ha insistito nel 2004 per avere una policy RF (Royalty Free), ciò spinse ad abbandonare il consorzio (o il working group) varie grosse imprese.
- WHAT: Nel 2004, Firefox e Opera proposero al W3C la riapertura del Working Group su HTML, ciò ignorava il rigido XML e venne respinto al W3C. Venne allora formato un gruppo separato, chiuso e finanziato dalle società di software, il Web Hypertext Application Technology, che sviluppò proposte (Web Application 1.0) che vennero effettivamente implementate da vari browser. Nel 2007 il W3C dovette ammettere che queste modifiche avevano un impatto innegabile, riaprì il working group su HTML con i membri del WHAT per creare una nuova versione: HTML 5.

Il World Wide Web è un sistema per la presentazione a schermo di documenti multimediali, e per l'utilizzo di link ipertestuali per la navigazione. Il sistema è distribuito e scalato su tutta Internet, ed è basato su alcuni semplici concetti:

- Il client (o browser) è un visualizzatore di documenti ipertestuali e multimediali.
- I browser hanno plug-in che permettono di visualizzare ogni tipo di formato speciale, e un linguaggio di programmazione interno (Javascript).
- Il server è un meccanismo di accesso a risorse locali in grado di trasmettere via socket TCP documenti individuati da un identificatore univoco (URI).
- Il server può collegarsi ad applicazioni server-side ed agire da tramite tra il browser e l'applicazione cosicché il browser diventa l'interfaccia dell'applicazione (es. SQL).

Alla base di WWW ci sono i seguenti protocolli:

- URI: uno standard per identificare in maniera generale risorse di rete e per poterle specificare all'interno di documenti ipertestuali.
- HTTP: un protocollo di comunicazione stateless e client-server per l'accesso a risorse ipertestuali via rete.
- HTML: un linguaggio per la realizzazione di documenti ipertestuali basato su SGML (per HTML e XML per XHTML) e incentrato sulla possibilità di realizzare connessioni ipertestuali in linea nella descrizione strutturale del documento.

Sito web statico

C'è un file per ogni schermata possibile, con un diverso URL, e nessun contenuto visualizzato cambia rispetto al documento memorizzato su disco.

Sito web dinamico

Parte dei contenuti è statica e memorizzata su file, ma la parte importante è generata in output da un applicazione server-side. Possono essere a tre livelli: browser - application logic (+ presentation) - storage, e di tipo embedded code o full application. Oppure a quattro livelli: browser - application logic - presentation - storage.

Rich client
Il lato presentativo viene spostato dal server al client, tramite javascript, framework correlati ed Ajax. Possibile spostare anche il lato application logic.

Caratteri
Prima di tutto per digitalizzare un alfabeto è necessario individuare un insieme di valori numerici da associare ai caratteri. Le regole principali per determinare questa associazione sono:
- Ordine: i valori numerici seguono l'ordine alfabetico culturalmente riconosciuto.
- Contiguità: ogni valore numerico compreso tra il più basso e il più alto è associato ad un carattere.
- Raggruppamento: l'appartenenza ad un gruppo logico è facilmente riconoscibile da considerazioni numeriche.
Da notare la presenza di codici di controllo, ovvero codici legati alla trasmissione e non al messaggio, o codici liberi, ovvero codici non associati a nessun carattere.

ASCII
L'American Standard Code for Information Interchange è uno standard ANSI che definisce valori per 128 caratteri, ovvero 7 bit su 8, in quanto il primo bit era considerato un bit di parità. Di questi vi sono 95 caratteri dell'alfabeto latino, maiuscole, minuscole, numeri e punteggiatura. Codifica contigua ed ordinata.

EBCDIC
L'Extended Binary Characters for Digital Interchange Code è una codifica proprietaria IBM a 8 bit (utilizzati tutti e 8, senza bit di parità).

ISO 646-1991
Questa codifica ISO è nata per permettere l'utilizzo di caratteri nazionali europei in un contesto ASCII. Lascia 12 codici liberi per le versioni nazionali dei vari linguaggi europei. Ogni tabella nazionale li usa per i propri fini.

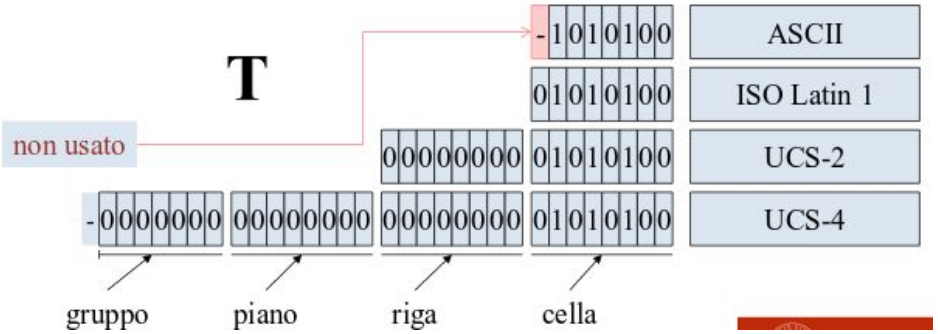
Code page ASCII
Con il miglioramento dell'hardware nella codifica ASCII il bit di parità è iniziato a risultare sempre più inutile. Si è deciso di sfruttare quindi i rimanenti 128 caratteri tramite un meccanismo di estensioni multiple ed indipendenti di ASCII. Tuttavia esistono centinaia di code page e non c'è alcun modo di sapere quale è stata utilizzata in un testo.

KOI7 e KOI8
Sono codifiche per i caratteri in cirillico realizzate dal governo sovietico a partire dal 1974.

CJK
È una codifica che raggruppa alfabeto cinese, giapponese e coreano. Essendo il numero di caratteri molto maggiore di 256 è stata usata una codifica a 16bit (56000).

ISO Latin 1
Estensione di ASCII fatta per utilizzare il primo bit (ed avere 256 caratteri in tutto). Unica estensione resa standard.

Unicode
Avere tante codifiche non facilità però la comunicazione, in quanto a seconda della codifica usata si possono avere dozzine di interpretazioni diverse per lo stesso codice numerico. Sono necessari quindi meccanismi indipendenti dal flusso per specificare il tipo di codifica usata, ad esempio tramite dichiarazioni esterne, intestazioni interne o un'interpretazione di default da parte dell'applicativo usato. Nascono Unicode e ISO/IEC 10646 che forniscono sostanzialmente gli stessi caratteri. Al momento definiscono 137.000 caratteri diversi, più 140.000 assegnati a scopi privati; sono divisi tra script moderni (latino, cirillico, …), script antichi (egiziano antico) e segni speciali (vedi emoji).
ISO 10646 è composto di due schemi di codifica:
- UCS-2 è uno schema a due byte. È un'estensione di ISO Latin 1.
- UCS-4 è uno schema a 31 bit in 4 byte, estensione di UCS-2. È diviso in gruppi, piani, righe e celle.

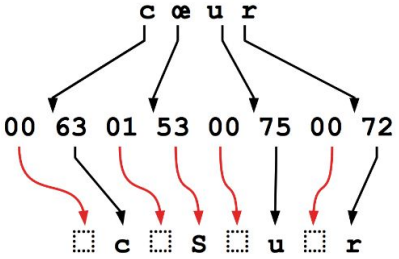


In UCS-4 esistono 32768 piani, il piano 0 è UCS-2 e comprende tutti gli alfabeti moderni, il piano 1 da tutti gli alfabeti antichi, il piano 2 caratteri orientali aggiuntivi, non già presenti.

UTF
L'Unicode Transformation Format o UCS Transformation Format) nasce dal fatto che è uno spreco utilizzare ogni volta 4 byte (UCS-4) quando un testo è scritto spesso in un solo alfabeto, o comunque utilizza una sola codifica. UTF-8 permette di accedere a tutti i caratteri definiti di UCS-4, ma utilizza un numero compreso tra 1 e 4 byte per farlo.
- I codici compresi tra 0-127 (ASCII a 7 bit), e richiedono un byte, in cui ci sia 0 al primo bit.
- I codici derivati dall'alfabeto latino e tutti gli script non-ideografici richiedono 2 byte.
- I codici ideografici (orientali) richiedono 3 byte.
- I codici dei piani alti richiedono 4 byte.
In generale, il primo byte contiene tanti 1 quanti sono i byte complessivi per il carattere (lunghezza).

0xxxxxxx	È un carattere UTF-8 completo da ASCII
10zyyyyx	È un byte di continuazione
110yyyyx	Primo byte di un carattere di uno script alfabetico
1110zzzz	Primo byte di un carattere di uno script ideografico
11110uuu	Primo byte di un carattere che non sta in BMP ma è già noto
111110uu	Non definito. Apparterrebbe ad un piano non ancora occupato

Problemi
Un problema può essere ad esempio quello degli accenti, per questi caratteri UTF-8 utilizza 2 byte, mentre ISO Latin 1 ne usa solo 1.



Content encoding
Molti ambienti informatici forniscono restrizioni sulla varietà di caratteri usabili, ad esempio modelli di rappresentazioni dei dati (stringhe nei linguaggi di programmazione, linguaggi di markup, …) o canali di trasmissione (protocolli internet).
Escaping: il carattere proibito viene preceduto o sostituito da una sequenza di caratteri speciali (String c = "questa \"parola\" escaped";).
Encoding: il carattere proibito viene rappresentato numericamente con il suo codice naturale secondo una sintassi speciale (“felicit\u00E0”).

MIME (Multipurpose Internet Mail Extensions)
Creato per bypassare i limiti all’interno di SMTP (troppo stringenti: caratteri solo ASCII a 7bit, max size 1Mb, …). MIME ridefinisce lo standard che regola gli header dei messaggi SMTP per permettere messaggi di testo in altri set di caratteri oltre ad US-ASCII, messaggi multi-parte, possibilità di messaggi non testuali.
Il flusso è il seguente: Il messaggio non compatibile con SMTP viene trasformato in uno o più messaggi SMTP da un preprocessore MIME nel server SMTP. Quello che viaggia su un canale SMTP è un puro messaggio SMTP con i limiti originari. All’arrivo, il o i messaggi SMTP vengono decodificati e riaccorpati da MIME per formare il messaggio originale.

Uniform Resource Identifier
Nel WWW ogni elemento di interesse è chiamato risorsa ed è identificato da un identificatore globale chiamato URI. Per definizione una risorsa è qualunque struttura che sia oggetto di scambio tra applicazioni all’interno del World Wide Web (quindi non necessariamente un file, ma anche un dato, un libro cartaceo, un concetto).
Gli Uniform Resource Identifier (URI) sono, per definizione:

- Uniform Resource Locator (URL): una sintassi che contiene informazioni immediatamente utilizzabili per accedere alla risorsa (ad esempio, il suo indirizzo di rete). Tuttavia sono fragili, e soggetti a modifiche (es. cambio del nome di una directory).
- Uniform Resource Names (URN): una sintassi che permetta una etichettatura permanente e non ripudiabile della risorsa, indipendentemente dal riportare informazioni sull'accesso. Questi devono però essere trasformati da un apposito servizio in URL associati alla risorsa, la mappa deve essere aggiornata ogni volta che viene spostata la risorsa.

```
URI = scheme : [// authority] path [? query] [# fragment]
authority = [userinfo@] host [: port]
```

I caratteri ammessi nell'URI possono essere:
- non riservati: uppercase | lowercase | digit | - _ ! ~ * ' ( )
- riservati: ; / ? : @ & = + $,
- escaped: %XX, con XX il codice esadecimale del carattere

URI reference
Un URI che riporta solo una parte dell'URI assoluto corrispondente, tagliando progressivamente cose da sinistra (es. pippo.png posto dentro a un documento fa riferimento alla stessa cartella in cui risiede).
Può avvenire URI resolution (da URI reference a URI assoluto) o URI dereference (da un URL alla risorsa).

Schema FILE
`file://host/path[#fragment]`
La parte host può essere eliminata, assumendo che sia localhost.
Schema DATA
`data:[<media type>][;base64],<data>`
Schema FTP
`ftp://[user[:password]@]host[:port]/path`

URL permanenti
In tutti i casi, un sistema server-side prende un URL permanente) a cui non corrisponde una risorsa fisica e identifica quale sia l'URL fisico corretto per quella risorsa.
URI rewriting
Permette la riscrittura dell'URI, facilitandone la comprensione e leggibilità (nascondendo inoltre dettagli sull'implementazione).
URI shortener
Esistono servizi che permettono di creare URL molto brevi corrispondenti ad URL molto lunghi, tale servizio effettua un semplice redirect.
IRI
IRI (Internationalized Resource Identifier), RFC 3987 di Gennaio 2005, fornisce una sintassi per inserire caratteri di UCS-4 in un URI e per risolverli. In realtà non tutti i caratteri sono ammessi: tutti i codici di controllo e tutti i codici non corrispondenti a caratteri usati nelle lingue non sono ammessi.

Linked Data

Si intendono tutte le risorse di dati che possono essere messi a disposizione ed incrociati da applicazioni ed umani.

HTML

Attualmente esistono due linee di sviluppo parallele di HTML:
- HTML 5.1 è una Recommendation del W3C
- HTML “Living Standard” del WHATWG

SGML

Lo Standard Generalized Markup Language è un meta-linguaggio non proprietario di markup descrittivo. Permette di definire markup leggibili, generici, strutturali, gerarchici.

HTML 4

Il linguaggio HTML è un tipo di documento SGML progettato per marcare documenti ipertestuali. Tramite HTML è possibile realizzare documenti con una semplice struttura, con aspetti grafici anche sofisticati, con testo, immagini, oggetti interattivi e link.
Nel tempo i browsers hanno permesso la visualizzazione di pagine HTML che non necessariamente rispettavano lo standard, da qui nascono il quirks mode (compatibile con il passato e più permissivo) e lo strict mode (compatibile con le specifiche ufficiali).

XHTML 1.0

Recommendation del 2000, in cui L’elenco e la semantica di elementi e attributi non è assolutamente cambiata rispetto a HTML 4. Qui i documenti debbono essere ben formati, in particolare l’annidamento deve essere corretto.

Nel 2004 venne creata una comunità aperta chiamata WHATWG (Web Hypertext Application Technology Working Group) finanziata e supportata da Mozilla, Opera e Apple, che ha iniziato a lavorare ad una versione “intermedia”di HTML. Nel 2007 visto l’impatto innegabile che aveva avuto, il W3C riapre il working group su HTML.
Nasce HTML “Living Standard”, il working group annuncia che non ci saranno standard rilasciati periodicamente, ma sarà in continua evoluzione.

- I tag fontstyle forniscono informazioni specifiche di rendering. Molti sono deprecati, e si suggerisce comunque sempre di usare gli stili CSS. (es. b, i, u, …)
- I font si suddividono in 5 famiglie: font con grazie (serif) font senza grazie (sans serif), font monospaziati, font calligrafici (script), font decorativi (fantasy).
- I tag di blocco definiscono l’esistenza di blocchi di testo che contengono elementi inline (es. p, div, blockquote, h1, h2, …)
- Gli elementi di lista sono ad esempio ul (unordered list) [li], ol (ordered list) [li], dl (definition list) [dt, dd].
- Elementi generici come div e span non hanno caratteristiche semantiche o presentazionali predefinite. Altri effetti sono hr (horizontal rule) e br (break).
- Gli attributi i18n garantiscono l’internazionalizzazione del linguaggio e la coesistenza di script diversi (es. lang e dir). Gli attributi di evento permettono di associare script

a particolari azioni sul documento e sui suoi elementi (onclick, ondoubleclick, onmouseover, onkeypress)

DOM
Il Document Object Model è un interfaccia di programmazione (API) per documenti sia HTML sia XML. Definisce la struttura logica dei documenti ed il modo in cui si accede e si manipola un documento. Utilizzando DOM i programmatori possono costruire documenti, navigare attraverso la loro struttura, e aggiungere, modificare o cancellare elementi. Ogni componente di un documento HTML o XML può essere letto, modificato, cancellato o aggiunto utilizzando il Document Object Model.
- Un DOMNode specifica i metodi per accedere a tutti gli elementi di un nodo di un documento, inclusi il nodo radice, il nodo documento, i nodi elemento, i nodi attributo, i nodi testo, ecc.
- Il DOMDocument specifica i metodi per accedere al documento principale, tutto compreso. È equivalente alla radice dell'albero (non all'elemento radice).
- Un DOMElement specifica i metodi e i membri per accedere a qualunque elemento del documento.

HTML 5
header: contiene l'intestazione della sezione corrente o dell'intera pagina ed è solitamente usato per tabelle di contenuti, indici, form di ricerca, intestazioni.
footer: contiene la "parte conclusiva" della sezione corrente o dell'intera pagina. È usato principalmente per mostrare informazioni (testuali, non metadati) sugli autori della pagina, copyright, produzione, licenze.
nav: particolari sezioni dedicate a raggruppare link alla pagina corrente (o a sezioni di) o ad altre pagine (valide per l'accessibilità).
canvas: definisce un'area rettangolare in cui disegnare direttamente immagini bidimensionali e modificarle in relazione a eventi, tramite funzioni Javascript.
video: specifica un meccanismo generico per il caricamento di file e stream video, più alcune proprietà DOM per controllarne l'esecuzione.
tipi di input: email, url, number, range, date, search, color.
localStorage: salvataggio nel browser di coppie chiave-valore.

HTTP e REST
HTTP é un protocollo client-server, generico e stateless. Con generico si intende che è indipendente dal formato dei dati che vengono trasmessi (binari, html, eseguibili, …). Con stateless si intende che non è tenuto a mantenere informazioni che persistano tra una connessione e quella successiva.

Importante la figura dell'user agent: un client che inizia la richiesta HTTP (un browser, un bot, …). L'origin server, è il server che possiede fisicamente la risorsa richiesta. Un proxy è un'applicazione intermedia che agisce sia da client che da server. Un gateway è un'applicazione che agisce da intermediario per qualche altro server, a differenza del proxy questo riceve le richieste come fosse l'origin server. Un tunnel è un programma intermediario che agisce da trasmettitore passivo di una richiesta HTTP.

La versione HTTP 1.1 ha due features aggiuntive: connessione persistente (la connessione viene mantenuta aperta fino a che è necessario), e pipelining (vengono trasmesse più richieste senza attendere l'arrivo della risposta ad ognuna).
La versione HTTP 2 ha ulteriori features: il multiplexing (in una connessione si possono avere risposte multiple che saranno ricostruite nell'host destinatario, aumentando il parallelismo), la compressione degli headers (il consumo di banda si riduce notevolmente) e operazioni di push da parte dei server (invio di dati ai client in modo da anticipare richieste future).

Metodi HTTP
I metodi hanno due proprietà: sicurezza (un metodo è sicuro se non genera cambiamenti allo stato interno del server, a parte i log) e idempotenza (un metodo è idempotente se l'effetto sul server di più richieste identiche è lo stesso di quello di una sola richiesta, a parte i log, un metodo può essere ri-eseguito da più agenti in tempi diversi senza effetti negativi).
- GET: recupera una rappresentazione di una risorsa [sicuro, idempotente].
- HEAD: recupera una rappresentazione delle informazioni che il server ha su una risorsa, ovvero restituisce solo gli header relativi, senza il corpo [sicuro, idempotente].
- POST: trasmette delle informazioni dal client al server (non creare nuove risorse) [non sicuro, non idempotente].
- PUT: trasmette delle informazioni dal client al server, creando o sostituendo la risorsa specificata [non sicuro, idempotente].
- DELETE: rimuove una risorsa ed ogni informazione presso di essa [non sicuro, idempotente]
- OPTIONS: fornisce l'elenco delle opzioni attive del server web, il suo scopo è verificare se un'eventuale richiesta futura verrà accettata non sulla base della risorsa richiesta, ma sulla base dell'identità del richiedente (utile per il cross-site scripting).

Gli Header
Gli header specificano 4 tipi di informazioni:
- la trasmissione (es. Date, Transfer-Encoding, Cache-Control, Connection)
- l'entità trasmessa (es. Content-Type, Content-Length, Content-Encoding)
- la richiesta effettuata (User-Agent, Referer, Host)
- la risposta generata

Lo *status code* è un numero di 3 cifre che indica l'esito della risposta. Quando il primo numero è 1 indica informazione, 2 indica successo, 3 indica redirection, 4 indica client error e 5 indica server error.

I cookies sono brevi informazioni scambiate tra il server e il client. Il client mantiene lo stato di precedenti connessioni e lo manda al server di pertinenza ogni volta che richiede un documento (questo perché HTTP è stateless). Alla prima richiesta di uno user-agent, il server fornisce la risposta ed un header aggiuntivo, il cookie, con dati arbitrari, e con la specifica di usarlo per ogni successiva richiesta. Il server associa a questi dati ad informazioni sulla transazione. Ogni volta che lo user-agent accederà a questo sito, rifornirà i dati opachi del cookie che permettono al server di reidentificare il richiedente.

Cross-Origin Resource Sharing (CORS)
Si attiva solo per le connessioni Ajax (non si usa per connessioni HTTP normali) e prevede l'uso di due nuovi header:
- nella richiesta: Origin, per specificare il dominio su cui si trova il contenuto
- nella risposta: Access-Control-Allow-Origin per indicare gli altri domini da cui è permesso caricare contenuti.
Inoltre si pone l'accento sull'uso di un preflight (verifica preliminare) della possibilità di eseguire un comando cross-scripting, ad esempio usando il metodo OPTIONS di HTTP.

REpresentional State Transfer (REST)
Un'applicazione REST si basa fondamentalmente sull'uso dei protocolli di trasporto (HTTP) e di naming (URI) per generare interfacce generiche di interazione con l'applicazione, e fortemente connesse con l'ambiente d'uso. Questa architettura si basa su 4 presupposti:
- definire risorsa ogni concetto rilevante dell'applicazione web
- associare un URI come identificatore e selettore primario
- esprimere ogni operazione dell'app web come creazione (PUT), visualizzazione (GET), cambio di stato (POST) e cancellazione (DELETE), modello CRUD.
- esprimere tramite parametri ogni rappresentazione dello stato interno della risorsa, attraverso Content Type (es. html, json, …)

CSS
Trasformazioni: translate (sposta la scatola), scale (cambia la dimensione della scatola), rotate (ruota la scatola), skew (inclina la scatola).

Animazioni: @keyframes (blocco di regole per specificare lo stato iniziale, intermedio e finale di una o più proprietà numeriche CSS).

L'ordine con cui vengono considerate le varie regole non è dato solo dall'ordine in cui sono posti nei fogli di stile. Infatti, viene applicato un algoritmo di ordinamento sulle dichiarazioni secondo alcuni principi (dal più al meno importante): media-type, importanza (!important), origine, specificità, ordine.

Attraverso preprocessori CSS è possibile estendere il CSS tramite compilazione o interpretazione. Due preprocessori noti sono LESS e SASS, che forniscono variabili, mix-in (tipo funzioni) e aritmetica.

Semantic Web
Vi sono molti modi per riutilizzare gli URI, ad esempio attraverso: il riutilizzo *diretto*, il *sameAs* (OWL) o il *seeAlso* (RDFS).
Il linguaggio utilizzato per il semantic web è RDF (Resource Description Framework). Questo linguaggio si basa su triple RDF: **subject** (una risorsa, che può essere identificata da un URI), **predicate** (una specificazione della relazione, identificata da un URI), **object** (una risorsa o un letterale al quale il subject è collegato).

Es.: <http://musicbrainz.or/artist/abcde> <http://xmlns.com/foaf/0.1/name> “The Beatles” .

RDF Turtle
È una sintassi per RDF più leggibile. Dato che vari URI condividono le stesse basi, si possono utilizzare prefissi:
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:<http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:<http://www.w3.org/2002/07/owl#> .
@prefix mo:<http://purl.org/ontology/mo/> .
@prefix dbpedia:<http://dbpedia.org/resouce/> .
@base <http://musicbrainz.org/artist/> .

<artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#_> **a** mo:MusicGroup .
Qui **a** equivale a: <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>

Possono essere usati anche tag linguistici:
dbpedia:The_Beatles
dbpedia-owl:abstract
"The Beatles were an English rock band formed (...) "@en ,
"The Beatles waren eine britische Rockband in den (...) "@de .

RDF/XML
<rdf:Description
rdf:about="http://musicbrainz.org/artist/b10bbbfc-cf9e-42e0-be17-e2c3e1d2600d#_">
<foaf:name>The Beatles</foaf:name>
<owl:sameAs rdf:resource="http://dbpedia.org/resource/The_Beatles" />
</rdf:Description>

RDF Schema
Inferenza: da un insieme di asserzioni, utilizzare queste classi e proprietà, specificare cosa dovrebbe essere inferito su queste asserzioni che è stato implicitamente fatto.

rdf:Property - Classe di proprietà RDF, ad esempio *mo:member* indica un membro di un gruppo musicale.
rdf:domain - Indica che ogni risorsa con una certa proprietà è un'istanza di una o più classi. Ad esempio *mo:member rdfs:domain mo:MusicGroup*.
rdfs:range - Indica che i valori di una proprietà sono istanze di una o più classi. Ad esempio: *mo:member rdfs:range foaf:Agent*.

JSON-LD
Metodo introdotto per codificare Linked Data utilizzando JSON. JSON-LD specifica un numero di tokens sintattici e parole chiave:
- "**:**" il separatore per le chiavi JSON e i valori che utilizzano IRI compatti
- "**@context**" definisce i nomi abbreviati chiamati termini che sono utilizzati all'interno del documento JSON-LD
- "**@graph**" utilizzato per esprimere un grafo
- "**@id**" utilizzato per identificare univocamente cose che sono state descritte nel documento con IRI o identificatori di nodi vuoti

- “**@language**” utilizzato per specificare il linguaggio per una particolare stringa o il linguaggio di default per un documento JSON-LD
- “**@prefix**” se è impostato a true permette a questo termine di essere utilizzato per costruire un IRI compatto
- “**@type**” utilizzato per impostare il tipo di dato di un nodo o un valore tipato
- “**@value**” utilizzato per specificare i dati che sono associati con una particolare proprietà del grafo

```
{ "@context":
      { "name": "http://schema.org/name",
        "image": { "@id": "http://schema.org/image", "@type": "@id" },
        "homepage": { "@id": "http://schema.org/url", "@type": "@id" }
      },
      "name": "Manu Sporny",
      "homepage": "http://manu.sporny.org/",
      "image": http://manu.sporny.org/images/manu.png
}
```

SPARQL
Un linguaggio di query per RDF.
Alcuni sono noti nella community del Semantic web, ad esempio foaf (friend of a firent), descrizione di relazioni sociali, mo (music ontology) descrizione di artisti, album e canzoni, sioc (semantically interlinked online communities) descrizione di comunità online, …
Un RDF graph è un insieme di asserzioni RDF, manipolate come un grafo diretto. Un dataset RDF è un insieme di triple RDF composta un grado di default e 0 o più grafi nominati.

L’idea principale di SPARQL sono query che descrivono sotto-grafi del grafo interrogato. Ad esempio *dbpedia:The_Beatles foaf:made ?album* fornisce dei sottografi che combaciano con i modelli di grafo forniti come risultato.

I formati di interrogazione sono: ASK (testa se un query pattern ha una soluzione o no, true o false), SELECT (ritorna le variabili e i loro collegamenti), CONSTRUCT (ritorna un singolo grafo RDF specificato da un graph template), DESCRIBE (ritorna un singolo grafo RDF contenente dati RDF su una risorsa).

Javascript
ECMAScript è lo standard per un linguaggio di script client-side, standardizzato da ECMA.
I dati in Javascript sono tipati, mentre le variabili no. Con **var** si definisce una variabile nello scope della funzione o del file in cui si trova, con **let** si definisce una variabile nello scope del blocco parentetico della variabile in cui si trova. Il confronto può avvenire con o senza casting dei valori (== con casting, === senza casting).

window: oggetto top-level con le proprietà e i metodi della finestra principale.
navigator: oggetto con le proprietà del client (nome, versione, plug-in installati, info, …).
location: URL del documento corrente.
history: array degli URL acceduti durante la navigazione.

document: contenuto del documento, ha proprietà e metodi per accedere ad ogni elemento.
Manipolare il DOM
innerHTML: legge/scrive il contenuto di un sottoalbero (escluso il tag dell'elemento)
outerHTML: legge/scrive il contenuto di un elemento (incluso il tag dell'elemento)
Alcuni selettori sono: *getElementById*, *getElementsByName*, *getElementsByTagName*, in aggiunta si ha *getElementsByClassName*, *querySelector*, *querySelectorAll*.

AJAX
Asynchronous JavaScript And XML è una tecnica per la creazione di applicazioni web interattive, e permette l'aggiornamento asincrono di porzioni di pagine HTML.
Non è un linguaggio o una tecnologia specifica, è una combinazione di tecnologie utilizzate sul web:
- XHTML/CSS,
- manipolazione dinamica del DOM tramite JavaScript,
- XMLHttpRequest (XHR) per lo scambio di messaggi asincroni,
- XML/JSON come meta-linguaggi per i dati scambiati.

Lettura e visualizzazione dati con AJAX

```
myXMLHTTPRequest = new XMLHttpRequest();
myXMLHTTPRequest.onreadystatechange = showData;
myXMLHTTPRequest.open("GET", "example1.xml", true);
myXMLHTTPRequest.send(null);

function showData() {
      if(myXMLHTTPRequest.readyState == 4) {
            if(myXMLHTTPRequest.status == 200) {
                  //legge la risposta
                  xmlDoc = myXMLHTTPRequest.responseXML;
                  //recupera il frammento
                  fragment = xmlDoc.getElementbyId("content").item(0);
                  // modifica il documento corrente
                  document.getElementById("area1").
                  appendChild(fragment);
            }
      }
}
```

Valori falsy e valori truthy
falsy: false, 0, null, undefined, "", NaN
truthy: "value", {}, [], "0", "undefined", "null"

Funzioni
In JS le funzioni sono oggetti, che oltre ai classici oggetti possono anche essere invocate con l'operatore (). Possono essere assegnate a variabili e possono avere 0 o più parametri. Si possono restituire funzioni come risultato di altre funzioni, creare funzioni dentro a funzioni, creare funzioni anonime come parametri di funzione.

Classi

Le classi non sono entità di primo livello, si usano piuttosto oggetti che provengono dallo stesso costruttore. Un costruttore è banalmente una funzione che restituisce un oggetto, si una new per usarlo come costruttore dell'oggetto.

```
function Persona(nome,altezza,nascita){  // Costruttore
   this.nome = nome
   this.altezza = altezza
   this.nascita = nascita
   this.saluta = function() { return 'ciao!' }
}
```

Nei linguaggi prototype-based non esiste il concetto di classe, ma quello di prototipo, un'istanza primaria, astratta, sempre accessibile e modificabile, di cui le singole istanze clonano (ed eventualmente modificano) sia membri che metodi. Esempio:

```
Persona.prototype.welcome = function(){
      alert("Benvenuto, " + this.name + "!");
}
```

Un modo intelligente di utilizzare prototype è quello di modificare il prototype della classe built-in su cui si stanno facendo molti controlli, aggiungendo una funzione che racchiuda tutti questi controlli, per poi utilizzarla sempre.

Closure

Javascript non ha protezione dei membri privati di un oggetto, ma sono tutti accessibili e manipolabili. Abbiamo detto che in Javascript ci sono solo due scope: quello globale e quello della funzione. Non è vero, c'è un terzo scope, detto closure, che è lo scope della funzione all'interno della quale viene definita la funzione. Ad esempio, una funzione che restituisce una funzione ha uno scope che è sempre accessibile alla funzione interna, ma non dal mondo esterno.

```
Counter = function() {
  var state = 0;                    // privata
  return {
    inc: function() { return ++state },
    dec: function() { return --state }
  }
}
var c = new Counter() ;
c.inc() ; var d = c.inc()  // d vale 2, corretto.
c.state = 7 ; var e = c.inc()  // e vale 3, c.state
è lecita ma inutile.
```

IIFE (Immediately Invoked Function Expression)

Una function expression immediatamente invocata (IIFE) è una funzione anonima creata ed immediatamente invocata. Serve sostanzialmente per fare singleton (oggetti non ripetibili) dotati di closure (e quindi di stato interno privato).

```
var people = (function { return help: function() {console.log("help")}})()
```

Asincronicità

Abbiamo esigenze di asincronicità ogni volta che abbiamo l'esigenza di chiamare un servizio sulla cui disponibilità o sui cui tempi di esecuzione non abbiamo controllo.

Generator/yield
Il generatore è una meta-funzione (una funzione che restituisce una funzione che può essere chiamata ripetutamente ed interrotta fino a che ne hai nuovamente bisogno). Ha una sintassi particolare con * dopo function. Il comando yield mette in attesa la assegnazione di valore fino a che non si chiude l'esecuzione della funzione chiamata. La funzione next() fa proseguire l'esecuzione della funzione fino al prossimo yield.

EcmaScript 2015
Funzioni inline:
```
var rect = arr.map(function(a, b) { return a * b });
```
diventa
```
var squares = arr.map ((a, b) => a * b );
```
Stringhe:
```
var firstName = 'Jane';
var x = `Hello ${firstName}!`
```
Gli oggetti sono ancora basati su prototipi, ma cambia la definizione, utilizzando la keyword *class*.

Accessibilità
Screen reader - application mode (modalità utilizzata nelle applicazioni native dei sistemi, abbastanza limitata alle API di accessibilità fornite dall'app).
Screen reader - document mode (per agevolare la lettura di documenti complessi, più sofisticata).

Frameworks
Le caratteristiche comuni ad ogni framework AJAX sono: l'accesso al DOM, comunicazioni asincrone con il server, una gestione degli eventi e una libreria di widget.
Un Content Delivery Network (CDN) è una rete fortemente distribuita di computer che collaborano tra loro per distribuire in maniera omogenea contenuti di grande successo senza inutili duplicazioni di occupazione e trasmissione di file.

jQuery
Tutti gli script vengono eseguiti subito dopo il loro caricamento. Ci sono due momenti specifici a partire dai quali si possono attivare gli script: *document.ready* (il DOM è caricato, pronto e inizializzato, ma alcune risorse potrebbero non essere ancora state caricate), *window.load* (il DOM è caricato e pronto, e tutte le risorse esterne sono pronte e caricate. Alcune funzioni per la comunicazione asincrona sono $.ajax(), $.get(), $.post(), $.put(). Il selettore $() è una promessa, quindi per le funzioni sopra si possono utilizzare le funzioni then().

MEAN stack
MongoDB, ExpressJs, AngularJs, NodeJs. Un'applicazione NodeJs è un'applicazione nativa server side che permette di fare chiamate I/O guidate da eventi, non-bloccanti. Invece di dedicare un thread ad ogni connessione ricevuta dalla rete, l'applicazione NodeJS utilizza un solo thread in cui tutte le richieste ricevono attenzione condivisa. NodeJS può porsi in

ascolto su qualunque porta e implementare anche protocolli diversi da HTTP. Node.js è quasi esclusivamente basato su funzioni asincrone e callback.

Search Engine Optimization (SEO)
Ci sono due tipi di risultati: ricerca organica (elementi ritornati dall'algoritmo del motore di ricerca) e ricerca sponsorizzata (es esempio le pubblicità). Un motore di ricerca è composto da tre processi: un crawler, un indexer e un query engine. Un crawler, conosciuto anche come (web) spider, è un internet bot che naviga nel WWW con lo scopo di indicizzare il web Esistono miliardi di pagine Web, pertanto il crawler deve essere costantemente eseguito su un numero elevato di computer. Il crawler prende gli URL dalle ricerche di indicizzazioni precedenti e dalle sitemap XML, quindi cerca di trovare pagine nuove o aggiornate e di aggiungerle all'indice di Google (rileva i collegamenti SRC e HREF).
I criteri che interessano al crawler sono: quanto impiega a caricare una pagina, quanto è importante il suo URL e se ci sono altri link su una pagina.
È possibile indicare nel proprio sito web, quali pagine si vuole indicizzare e quali no (tramite un file chiamato robots.txt).

Javascript è un problema per i crawler, subentrano allora gli indexers. Quando un crawler processa una pagine, cerca degli hyperlink, li manda all'indexer che li interpreta ed esegue il javascript, che spesso produce altri URL. che sono ritornati al crawler. Il processo termina quando il crawler non può andare più oltre.

Un PageRank rappresenta un algoritmo matematico utilizzato per determinare l'importanza di una pagina, e tale processo è essenzialmente basato sulla valutazione della quantità e qualità di link che conducono a quella specifica pagina.

Il concetto base per l'indicizzazione di Javascript è: se è richiesta l'azione di un utente per caricare dei contenuti, allora quel contenuto non sarà indicizzato.