03/10/2022: Presentazione del corso e contenuti

Creare un sito web intende creare un prodotto per tutti, cioè un utente generico.

Manca di base una vera cultura: infatti, tanti prodotti sono pensati lato desktop in primis, ma oggi la maggior parte degli accessi è mobile.

*Contenuti del corso*

* Il concetto di ipertesto, il World Wide Web ed Internet. Gli enti di standardizzazione, le architetture Client-Server e i protocolli di Internet.
* I linguaggi del web statico: XHTML, HTML5, CSS.
* Web design. Paradigmi di progettazione Mobile. Emotional Design.
* Accessibilità & SEO (Search Engine Optimization)
* I linguaggi per il web dinamico: Javascript, PHP.

Cose utili:

* Le lezioni teoriche sono registrate
* Il progetto conta per il 75% del voto, lo scritto il 25% (se non soddisfatti, si può chiedere un orale)
  + 3 Consegne del progetto (sono indipendenti scritto e progetto, ma per il 30 settembre 2023 si devono superare o si perdono)
  + Per chi consegna il progetto per la prima sessione, ci sono dei punti bonus (consegna per inizio febbraio)
* Il progetto è fatto da 4 persone e si realizza un sito web
* Il progetto deve essere *politically correct* (temi sensati concettualmente)
* I laboratori non vengono registrati e occorre iscriversi al Moodle entro il 10 ottobre 🡪 Meglio farseli, le soluzioni non vengono diffuse e non si finisce mezz’ora prima per i laboratori
* Progetto e scritto sono corretti dopo la verbalizzazione di Basi di Dati (comunque, si ha propedeuticità anche rispetto a Programmazione 1)
* Usiamo Wooclap per alcune domandine sia in aula che da casa

Viene fatto un parallelismo tra la tecnica di costruzione architetturale e il Web: in poco meno di un secolo, si ha avuto un’evoluzione velocissima, sia da un punto di vista progettuale che realizzativo.

Dal punto di vista di Web, si deve pensare:

* cosa vuole l’utente
* chi è l’utente (il campione medio è il maschio bianco caucasico)
* dove viene visto il sito Web e come viene raggiunto (si intende da che piattaforma, es. desktop, assistenti vocali, mobile, applicazioni, browser interni ad app terze, ecc.)

Esiste un *divario digitale/Digital Divide* da un punto di vista di accesso alle risorse e si può subire esclusione per cause dovute all’istruzione, al sesso, condizioni economiche-sociali, età, ecc.

Similmente, parliamo di “utenti svantaggiati”:

* gli anziani
* chi non intende la lingua (in un senso di comunicazione)
* chi non intende il gergo (che non sia troppo tecnico)

Essi sfruttano dei mezzi di I/O alternativi (si danno alcuni esempi a puro titolo esemplificativo e didattico):

* tastiere e display Braille, quindi usabili da utenti non vedenti o ipovedenti
* microfoni e tastiere virtuali in applicazioni specifiche
* head-mounted pointers/puntatori laser, per permettere il puntamento e l’interazione classica tramite movimenti fisici tramite movimento oculare, espressioni facciali, ecc.
* puff-suck switch, usando un tubo connesso tramite cavo al PC per interagirvi

Eventuale approfondimento per questo a:

<https://quizlet.com/75163048/input-and-output-devices-for-disabled-users-flash-cards/>

Per questi motivi, l’accessibilità è fondamentale. Per esempio, l’accesso alle pagine testuali (quindi, una pagina può essere, escludendo contenuti come immagini, video ed effetti estetici) deve essere garantito, sia per un discorso di accessibilità che di indicizzazione diretta da parte di Google.

In assenza di questo, il classico *text and speech* (quindi, il riconoscimento del testo di una pagina, interpretato da parte di assistenti vocali e descritto vocalmente da questi, utile per utenti non vedenti ad esempio) potrebbe non funzionare e non si rende, a pieno, il sito accessibile, né tantomeno interpretabile o traducibile.

Un altro problema è *l’attesa* del sito vero e proprio, particolarmente rilevante per persone disabili.

* Problema classico: Immagine pesante e non compressa sulla homepage; ci possono volere anche alcuni secondi per caricarla direttamente
* Ci vogliono normalmente 3/4 clic/tap per riuscire a trovare l’informazione (limite massimo accettabile per l’utente; oltre questi, potrebbe non tornare più sul nostro sito). Se all’utente interessano informazioni particolari, il numero sale fino a 7, tuttavia è un caso più limitato

Fondamentale anche il testing:

* Tutti i link devono essere accessibili (quindi, essere cliccabili e condurre a pagine esistenti e non lanciare un generico errore 404, cioè “page not found – pagina non trovata”)
* Fare dei test su ogni componente del sito
  + Esempio: Tag e codice valido (i tag HTML, cioè delle componenti usate nelle pagine web che servono a descrivere gerarchicamente i contenuti estetici e di forma della pagina, devono essere chiusi, il codice deve essere correttamente espresso e definito compilando, ecc.)
  + Esempio: Codice senza commenti e senza alcuna spiegazione sul funzionamento (risulta complicato sia per chi lo ha scritto ad una successiva rilettura sia per future reinterpretazioni da parte di altre persone)
* Quale tra le ultime due soluzioni è migliore? *Dipende*
  + In alcuni casi, serve a seconda del linguaggio (se avere accessibilità e velocità e viceversa)

05/10/2022: Introduzione e linguaggio HTML

All’inizio, i browser cercavano, per quanto possibile, di interpretare il codice Web delle pagine da visualizzare. Per questo motivo, dato che questi cercavano di interpretare la corretta istruzione oppure l’istruzione corretta più prossima a quella corretta, si aveva una difficoltà interpretativa.

Questa differenza sussiste ancora nell’interpretazione attuale (quindi, un browser interpreta in un modo, un altro browser lo interpreta in un altro, ecc.).

Come tutti sappiamo, nasce per scopi militari e da scopi di connessione e ricerca di argomenti interconnessi tra di loro, si ha la rete Internet per come lo conosciamo.

Questa è l’idea di *ipertesto*, dunque accedere a risorse interconnesse le une con le altre.

Il vero inizio della rete si ha ad inizio anni Novanta con HTML, che collegava inizialmente risorse logicamente collegate tra di loro tramite l’ipertesto. Oggi si è quasi completamente collegati, anche in luoghi remoti o sconosciuti.

Per dare un’idea statistica: <https://www.broadbandsearch.net/blog/internet-statistics>

Similmente, gli accessi principalmente sono da telefono e, molto spesso, non con versioni recenti installate e questo va assolutamente considerato.

La comunicazione di Internet si basa su un insieme di reti interconnesse od eterogenee, basate sul protocollo TCP/IP, considerato *affidabile ma non efficiente* (in quanto, si sa che tendono a funzionare bene, ma non si sa con esattezza la vera tempistica). La stessa Internet intende INTERconnected NETworks, quindi, le risorse ci sono e potrebbero non arrivare oppure potrebbero non avere grandi velocità (*best effort*).

Esempio: JQuery, libreria particolarmente usata in ambito web, collegata direttamente col server ISP (del provider) di JQuery. Se non disponibile per qualche motivo, non si accede alla rete (idealmente, si può metterlo in cache e risolvere il problema).

Immagine che contiene tavolo

Descrizione generata automaticamenteImportante ricordare che si ha a che fare con un’architettura distribuita e soprattutto distinguere tra Internet e Web:

* Internet è un’infrastruttura tecnologica che permette di far comunicare diversi computer (la rete fisica)
* Il Web è l’insieme di software e protocolli installati sui diversi computer. In senso astratto è un insieme di documenti collegati tra loro.

Il modello utilizzato in rete è il modello OSI (pezzo a sx dell’immagine), corrispondente al TCP/IP nella realtà della rete (pezzo a dx dell’immagine); noi ci concentriamo sul livello Application del modello teorico ISO/OSI, corrispondente al caso nostro ad un insieme di protocolli di comunicazione interconnessi tra di loro. Se si volesse verificare e approfondire:

<https://vitolavecchia.altervista.org/differenza-tra-modello-tcp-ip-e-iso-osi/>

Il protocollo HTTP (HyperText Transfer Protocol) nasce nel 1999 per lo scambio di documenti tra client e server connessi tra di loro, in cui ciascun documento è identificato univocamente da URL. Per esempio, per quanto riguarda la *richiesta del client*:

Immagine che contiene testo

Descrizione generata automaticamente

Per quanto riguarda la *risposta da parte del server* Web, la richiesta viene ricevuta e poi renderizzata (il server Web ne riceve molte contemporaneamente), con una risposta di questo tipo:

Immagine che contiene tavolo

Descrizione generata automaticamenteSi consideri che i numeri intendono:

* 100
* 200
* 300
* 400
  + 404 (lato client, risorsa non trovata)
  + 405 (lato server, metodo non permesso)
* 500
  + 505 (risorsa eseguita e lato server si ha errore di compilazione)

Riferimento completo:

<https://umbraco.com/knowledge-base/http-status-codes/>

Per la risposta da parte del server Web, la modalità descritta prima si applica in caso di pagine statiche. Una pagina, o parte di essa, potrebbe essere dinamica: in questo caso il server crea “al volo” la pagina richiesta dall’utente sulla base dei dati forniti.

Il World Wide Web Consortium (W3C) è un organismo indipendente che comprende le maggiori ditte produttrici di software per la rete (es. Google, Intel, AOL, Apple, Microsoft, ecc.).

Si occupa di proporre standard a largo spettro che comprendono un gran numero di tecnologie ed iniziative che riguardano il Web (HTML, XHTML (questo combina XML e HTML), CSS, WAI (web accessibile), ecc.).

Le proposte (*Candidate Recommendation* o *working* *draft*, bozze progettuali) vengono messe a disposizione su Web per raccogliere il numero più alto possibile di contributi. Successivamente diventano standard (*Recommendation*). Prima erano standard semestrali, per dare il tempo agli utenti di adattarsi, poi i margini si sono notevolmente ridotti.

Sito di riferimento: <http://www.w3c.org/>

In genere mette a disposizione:

* la definizione dello standard: *recommendation*
* una suite di test per l’implementazione (*testsuite*)
* un servizio di validazione (usato come sorta di compilatore per verificare il funzionamento)
  + Esempio caso HTML5: <http://html5test.com/>

È tutto offerto in modo gratuito proprio per promulgare l’utilizzo e la diffusione dello standard in oggetto.

Per esempio, nel caso dello standard HTML: <https://html.spec.whatwg.org/multipage/>

Si considera che tutto rimanga retrocompatibile; è il caso di HTML5, divenuto lo standard de facto da alcuni anni. Si può vedere l’evoluzione dei singoli standard al link: <https://www.educba.com/versions-of-html/>

Importante sapere che HTML è linguaggio di *markup/annotazione*, in quanto si marca un documento per definire delle caratteristiche estetiche oppure tipografiche (non caratteristiche logiche, come fanno i linguaggi di programmazione).

All’inizio i designer provenivano dall’editoria cartacea:

* un output cartaceo fisso: un giornale può essere pensato come un file PDF (con qualche distinguo).

Nell’editoria su web entrano in gioco nuove variabili

* sistema operativo (tipi MIME (cioè, l’estensione di file, documenti o byte inviati in rete) e font supportati, etc. )
* caratteristiche del dispositivo (schermo, connessione, etc)
* browser (standard supportati, bug)

Essendo che non si sapeva come apparisse effettivamente il risultato finale grafico, si progettava in modo *ignoto (Far Web)*; inoltre, non esisteva uno standard ben definito (quali tag usare, come allineare, ecc.).

L’idea del design tradizionale pone dei dogmi (regole fisse), da dover definire con i grafici (mockup, realizzazione del sito senza la vera comprensione delle regole)

* Inalterabilità del carattere tipografico, usare un font fisso per tutte le pagine
  + Cambia in realtà da dispositivo a dispositivo, dalle dimensioni, dal browser, ecc.
* Inalterabilità del colore
* Inalterabilità della composizione
  + Design “orizzontale” del sito su desktop, design “verticale” su mobile
* Variabilità del modello e della versione del browser
* Variabilità del dispositivo utilizzato
* Velocità di connessione
* Preferenze utente (normalmente salvate come *cookies*)
* Caratteri, colori e composizione variabile

Non sempre le statistiche sono bilanciate, come ad esempio il sito W3Schools, utile per consultazione di tag web generici. Infatti, al sito: <https://www.w3schools.com/browsers/browsers_os.asp>

si vede un netto sbilanciamento verso Windows e desktop.

Ovviamente, i browser sono molti e bisogna considerarli tutti (dato che, in un browser una pagina potrebbe essere vista in un modo, mentre su un altro browser potrebbe non essere visualizzata e cose di questo tipo). Si intende, a titolo di chiarezza, un *browser* come un programma in grado di acquisire e navigare tra risorse Web e risorse locali. Esempi di browser sono: Chrome, Firefox, Safari, Opera, Edge, ecc.

Globalmente (al 2022) il più usato è Chrome, seguito nell’ordine da Edge, Firefox, Safari ed Opera.

Non è più un problema la risoluzione troppo bassa, ma al contrario quella troppo alta (da telefono, si vede tutto troppo ingrandito oppure la navigazione risulta faticosa a livello cervicale se i siti sono troppo grandi in larghezza). In questo, si considera che l’hardware sia in grado di visualizzare tutti i colori possibili (*profondità del colore/color depth*, che descrive quanti colori sono disponibili per ogni pixel) ormai e, comunque, non si deve più considerare che gli utenti disattivino JavaScript, cosa che avveniva comunemente nelle prime iterazioni della rete. Come soluzione:

* Non pretendere di avere l’assoluto controllo sull’output finale (spesso, si cerca di fare del proprio meglio, ma sul Web non esiste una soluzione universale)
* Non cercare di ottenere un’uguaglianza pixel a pixel sui diversi dispositivi nelle diverse situazioni (dato che ogni browser). Preferire invece:
  + Design fluidi (quindi, la pagina web si deve adattare all’interfaccia/browser che la sta visualizzando in quel momento)
  + Accessibilità vs. design visuale accattivante (cercare il giusto compromesso tra le due cose e caratteristiche)

*Nel progetto, si usa sempre HTML5, che la prof ci insegna; ciononostante, si introduce anche XHTML, in quanto storicamente collegato.*

Ciò detto, HTML è un linguaggio di markup per la costruzione di ipertesti, la cui prima versione si ebbe nel 1992 e poi standardizzata nel 1994, quando si ebbe la prima guerra dei browser, i quali all’inizio non collaboravano come fanno oggi, essendo solo una pura competizione.

(es. Netscape usa il tag img per le immagini, Microsoft voleva usare obj per tutti gli oggetti, ecc.

Da approfondire: <https://thehistoryoftheweb.com/the-origin-of-the-img-tag/>)

Grazie ad HTML si ha avuto WYSIWYG (What You See Is What You Get), andando verso un editing sempre più facile, visuale e con più supporto di colori e sintassi, con l’evoluzione dei siti sempre più spinta ad una rincorsa alla standardizzazione, per vari problemi (spreco di banda, pagine non accessibili, copia del codice e tag, ecc.). All’inizio era molto povero di contenuti, poi con il tempo si è notevolmente arricchito. Esso è parte di SGML (Standard Generalized Markup Language), standard di definizione dei linguaggi markup.

Esempi famosi di dispute tra tag e browser (*prima guerra dei browser*, in cui Internet Explorer diventò popolare a scapito di Netscape essendo gratuito, aggiungendo le stesse features ed altre anche migliori, tra cui tag diversi per le immagini, testo lampeggiante, testo scorrevole, ecc.), ma generando grossi problemi di incompatibilità tra i singoli programmi.

La *seconda guerra dei browser* si ha con Firefox (in cui è confluito Netscape successivamente, browser di AOL che ha avuto per anni grande mercato), dove si produca codice corretto e i tag si aprano e chiudano correttamente. Si cerca di porre il più possibile la validità degli editor grafici e il supporto agli standard presenti (zoom del testo, document switch (interpretazione dei documenti tramite gli standard), aggiornamento dei browser, ecc.), andando verso miglioramenti software e pure hardware (video) e, soprattutto introducendo un progetto comune di sviluppo (WaSP, Web Standard Project), con l’obiettivo di creare codice valido agli editor.

Allo stesso modo, si spinge per l’introduzione di nuovi standard estetici comuni (XHTML, CSS, XML, ecc.)

06/10/2022: Continuazione lezione HTML/XHTML

La scrittura delle pagine Web passa dall’interpretazione al caso specifico fino alla generalizzazione tramite software di scrittura siti e senza particolari competenze (CMS, Content Management System).

Dal punto di vista della definizione dello standard, viene definito lo standard XHTML 1.0 per cercare di dare un cambio di rotta: non più l’interpretazione di codice anche non corretto, ma si comincia a pretendere un codice valido (tag chiusi, attributi sempre con un valore, apertura-chiusura tag nell’ordine corretto, ecc.).

XHTML fu una novità drammatica per i creatori di CMS, in quanto per loro fu una grossa complicazione, sia per motivi di produttività che per motivi di accessibilità, dato che questi software permettono di creare siti senza competenze di programmazione e, in generale, ora tutto cambiava con XHTML.

La seconda versione cercò di pulire un po’ quanto creato fino a quel momento, con la versione XHTML 2.0, (inserendo per esempio un link non con *<a href>* ma non *<link>* e cose che ostacolavano la retrocompatibilità, in quanto non supportate globalmente). Questa versione era pensata per palmari e cellulari, cercando di ridefinire eventi e chiamate oltre che inserimento di immagini. Essa venne ripudiata perché contenuto e aspetto non vengono considerati separatamente, portando ad una crescita disordinata e ha rappresentato una rivoluzione di markup troppo grossa per essere chiaramente usata.

In risposta a XHTML 2, venne creato HTML5, per creare qualcosa di più orientato alle applicazioni web.

Venne quindi creato WHATWG, Web Hypertext Application Technology Working Group, per la definizione di form e app web. Nel dubbio, “prima gli utenti, poi gli autori e gli implementatori di pagine web”. Essi crearono vari organi per la gestione di web app e form per il web, coordinati sempre da W3C.

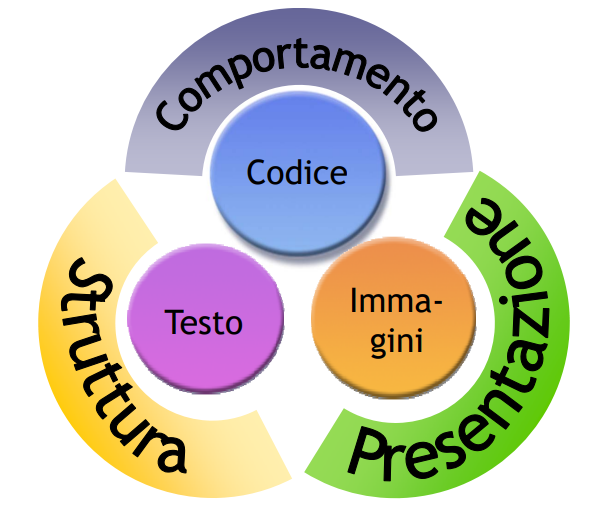
Per ogni possibile errore, si cerca di codificare ogni nuova situazione su un browser.

Al 2022, non esiste una gestione standard degli errori; tuttavia, si ha raggiunto il supporto completo da parte di tutti i browser almeno per quanto riguarda HTML5.

Per l’esame:

* Linguaggio: HTML5
* Grammatica: XML
* Verifica della validazione (costa, lo usiamo in laboratorio): Total Validator
  + Nel caso XHTML, serve W3C

Per l’HTML ci sono tre elementi fondamentali da separare completamente:

1. Struttura, testo che descrive come deve essere fatta la pagina
2. Comportamento, codice che qualifica come la struttura di un sito prende forma
3. Presentazione, tutto ciò che serve ad abbellire la pagina, da un punto di vista di immagini, layout, ecc.

Si hanno anche vari problemi:

* Crescita disordinata (incompatibilità)
* Contenuto e aspetto non vengono considerati separatamente (Pagine XHTML + fogli di stile CSS)
* Il numero notevole di pagine web presenti oggi rende difficile qualunque modifica al linguaggio HTML che non sia retrocompatibile

Il *code forking* è considerata bad practice, quindi cosa da non fare; seguendo l’accezione del termine alla GitHub, è un codice che viene scritto e ricopiato in modo uguale a quello di partenza, creando da esso un nuovo programma (concretamente, ad esempio, una serie di condizioni if per differenziare il comportamento delle pagine tra browser e per OS diversi; codice ripetuto e poco scalabile). Questo era un problema delle prime iterazioni di Chrome, che si falsificava come Internet Explorer per visualizzare correttamente i siti web.

XHTML è HTML riformulato come XML (riprende la versione 4.01 di HTML nella versione iniziale 1.0) quindi è più coerente e aiuta lo sviluppo di codice valido, eliminando parte dei problemi di presentazione di HTML. Essendo un linguaggio XML è interoperabile ed è supportato dai vecchi browser. Elimina il problema citato sopra di *code forking* in quanto supportato di per sé da diversi dispositivi (browser, browser per dispositivi mobili, screen reader, ecc). L’uso degli standard Web porta i seguenti vantaggi:

* Compatibilità con i browser (anche cambiandoli, si ha un design solido)
* Compatibilità con le future tecnologie (adattandosi ad uno standard prima, è facile evolvere poi con nuovi standard)
* Controllo centralizzato della presentazione (quindi, modificabile in maniera precisa alla bisogna sapendo che, partendo da regole definite, si può modificare facilmente (fatto bene prima vuol dire fatto bene poi))
* Indipendenza dal dispositivo (sempre conseguenza del seguire gli standard, quindi a prescindere dal dispositivo, si ha un chiaro risultato)
* Migliore posizionamento nei motori di ricerca (seguire le regole significa avere maggiore preferenza da parte degli algoritmi dei motori)
* Pagine leggere (si caricano in poco tempo e l’esperienza utente è migliore)
* Accessibilità (non creare un prodotto “peggiore” da un punto di vista grafico, ma creare un prodotto adattabile a varie tipologie di utenti)
* Migliore posizionamento sul mercato come sviluppatore web (tutto ciò migliora la vendita, la realizzazione e la reputazione)

Intero riferimento al link: <https://www.w3.org/TR/xhtml1/>

Tra le versioni di XHTML ricordiamo:

* *XHTML Transitional* è una forma transitoria creata per facilitare agli sviluppatori il passaggio ai nuovi standard
* *XHTML Strict* è la forma più pura che aiuta a produrre codice in cui struttura e presentazione sono fortemente separati. Per contro, non sempre è supportato bene dai vecchi browser
* *XHTML Frameset,* dove in pratica i tag contengono una serie di frame (da cui frameset), i quali potevano incorporare al loro interno numero pagine web HTML.

Diciamo che, progressivamente, si è cercato di portare un’introduzione di nuove tecnologie e tag per scopi di embedding (uno dentro l’altro), tali da poterli indicizzare se ben realizzati.

XHTML è un linguaggio XML quindi:

* i tag e gli attributi sono *case sensitive* (vanno sempre scritti in minuscolo e si intende letteralmente in italiano “sensibilità al maiuscolo”)
* i tag devono sempre essere chiusi (anche se sono vuoti)
  + *<br />* e non *<br>* (*break,* per andare a capo)
  + per compatibilità con i vecchi browser va usata la forma *<p></p>* (*paragraph,* che sta per paragrafo) per i tag non vuoti (anche se privi di contenuto) e *<br />* per gli elementi vuoti
* i tag devono essere aperti e chiusi nell’ordine corretto
* l’ordine con cui si inseriscono gli attributi è irrilevante
* i valori degli attributi vanno riportati tra “doppie virgolette” (quindi, come appena visto)
* tutti gli attributi devono avere un valore (di solito, un numero)
* un elemento in linea (non si avvia una nuova riga, normalmente *<span>, <a>, <img>,* occupando solo la larghezza necessaria) non può contenere un elemento di blocco (inizia sempre un’intera riga e occupa tutta la larghezza disponibile, ad esempio un *<div>*). In altri termini, non è possibile che un elemento contenga il proprio contrario (vedremo meglio entrambi questi elementi, citati per chi ha un minimo di conoscenza degli esempi di tag presentati).

I browser cercano di visualizzare *al meglio* codice non valido, ma questo può portare ad interpretazioni arbitrarie (magari, non sempre corrette). Essi ignorano completamente:

* le interruzioni di linea non identificate con *<br/>* e non contenute in un tag *<pre>* (cioè, *preformatted text,* testo preformattato)
* tabulazioni e spazi multipli
* tag *<p>* nidificati
* tag sconosciuti
* commenti
  + ATTENZIONE: dentro un commento non è possibile inserire la stringa “--” (doppi trattini)

(a quanto pare, questa feature esiste per assicurare che XML sia compatibile con lo standard SGML)

*Perché li ignorano*? In questo modo, si evita incompatibilità di tag nuovi/sconosciuti, paragrafi non chiusi, commenti, tali da visualizzare pagine anche in browser più vecchi.

La struttura di base di un XHTML (Strict in questo caso) è come segue:

Immagine che contiene testo

Descrizione generata automaticamente

Note per entrambe le immagini:

* Si ha una definizione del tipo di documento (*dtd, Doctype),* introdotta nei primi tempi per compensare gli errori di HTML nella compatibilità nei primi anni; può essere omessa
* *xmlns* intende il namespace
* si ha una strutturazione racchiusa sempre tra *html* (che racchiude tutto il documento), *head* (che racchiude informazioni descrittive sul documento (metadati) normalmente non visualizzate), *body* (contenitore per tutti gli elementi visibili della pagina)

Invece, la struttura di base di un HTML5 è come segue:

