



<!DOCTYPE> e <HEAD>: chi ben comincia...

Introduzione

In questo articolo descriveremo gli elementi che appaiono in una pagina (x)html prima e all'interno della sezione <HEAD>. Spiegheremo la loro funzione e come vanno scelti. Parleremo del doctype, di come fornire importanti informazioni sul nostro sito tramite i metadati e di quali accorgimenti si possono già adottare in questa parte del codice per rendere il sito più accessibile.

Il <!DOCTYPE>

Il primo elemento che deve/dovrebbe comparire nel codice della nostra pagina web è il <!DOCTYPE> (Document Type Declaration) che serve a dichiarare quale versione HTML o XHTML abbiamo utilizzato per scriverla e quindi a quale DTD (Document Type Definition) facciamo riferimento. Per esempio quello qui sotto è il **doctype di HTML 4.01 transitional comprensivo di URL**:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"  
"http://www.w3.org/TR/html4/loose.dtd">
```

Esistono per l'(X)HTML tre distinte DTD: **strict**, **transitional** e **frameset**. Il primo è il più rigido, implica l'abbandono di tutti gli elementi ed attributi di presentazione che devono quindi essere affidati ai fogli di stile. Se ciò non è possibile (per le specifiche richieste del progetto o per una nostra decisione progettuale) si può utilizzare una DTD Transitional (loose.dtd) che invece consente (o meglio "tollerata") l'uso di questi attributi. La Frameset è identica alla Transitional, tranne che, come dice il nome, prevede l'uso di frame e dei relativi tag. Infine è da notare che il doctype può riportare o meno l'url dove reperire la DTD che potrebbe essere pubblica e online, ma anche privata e in locale. Quindi le combinazioni che il doctype può assumere sono svariate.

Facciamo un passo indietro nel tempo per essere più chiari. L'HTML è nato come derivazione dell' SGML, un linguaggio standard ISO piuttosto complesso che comporta necessariamente la conformità con una DTD, ovvero con un documento (spesso esterno) che, definendo i <TAG> e la loro sintassi, specifica esattamente ciò che posso/devo/non-posso scrivere nel mio codice. L'HTML non è però uno standard e un doctype non è obbligatorio.

Diversa è la questione dell'XHTML. Essendo quest'ultimo la riformulazione in XML di HTML (ed anche il suo successore), un documento scritto con questo linguaggio punta ad essere valido e ben formato secondo la DTD che viene specificata nel doctype obbligatorio.

Questa informazione è utilissima in due modi:

1) Controllo del codice da parte dei validatori

I validatori (come questo [del W3C](#)) solo in questo modo possono dirci quali errori abbiamo commesso nello scrivere il nostro codice perchè altrimenti non potrebbero conoscere quale versione dell' (X)HTML abbiamo adottato. Preciso: non che a noi "esseri umani" per scrivere codice HTML occorra conoscere nella sua forma originale il [file DTD](#), ma sappiate che le [specifiche](#) di tale

linguaggio ricalcano queste DTD.

2)Rendering della pagina da parte dei browser

Netscape ed Explorer hanno implementato il supporto agli standard in modo differente. Netscape (che dalla versione 6 si appoggia al progetto open-source Mozilla e al suo motore Gecko, e di cui la 7 sarà l'ultima versione) ha fatto del supporto ai CSS e al DOM un suo vanto; Explorer invece ha implementato gradualmente dalla versione 5, attraverso la 5.5 e infine con la 6 (parliamo di versioni per PC), tale supporto. Per sopperire a queste differenze si è pensato di dare agli sviluppatori uno strumento in più: la possibilità di utilizzare il **doctype switch**. Attraverso il doctype si possono dare istruzioni ai browser su come devono comportarsi nel visualizzare la pagina. Tale possibilità viene introdotta da Internet Explorer 5 per Mac, Internet Explorer 6 PC, Mozilla, Netscape 6, tutti i browser basati su Gecko (come Camino, Galeon, Phoenix ecc.ecc.), Opera 7 e Safari.

Le tre modalità di funzionamento dei browser sono:

- **Quirk**: la pagina viene visualizzata come nelle vecchie versioni dei browser, in modalità non standard. Si garantisce così la retrocompatibilità.
- **Standard**: massimo supporto agli standard.
- **Almost standard**: standard, ma con una eccezione.

Quest'ultima modalità, disponibile solo con Mozilla (dall' 1.1) e Netscape 7, è uguale alla standard tranne che nella gestione del valore di line-height. Questo, sebbene implementato dai CSS2, in Explorer viene calcolato diversamente. Rispettando gli standard, Mozilla e Netscape mostrerebbero in confronto ad Internet Explorer un margine inferiore vuoto alla base delle immagini (se queste sono state allineate disponendole in celle di tabella, il problema diviene molto evidente). Per non avere tale diversità di visualizzazione Mozilla e Netscape hanno scelto di poter gestire **solo** questa regola come se si fosse in Quirk mode.

Alcuni dei comportamenti assunti dai browser in modalità Quirk sono molto noti dai webdesigner (e lo sono anche le "contromisure" come i [boxmodel hacks di IE5](#)), altri meno anche se documentati sui siti ufficiali dei browser. Fate attenzione però che i browser si "mettono" in modalità Quirk anche per motivi diversi da quello di una scelta precisa dello sviluppatore. Può accadere ad esempio per la presenza/mancanza della URL nel doctype, oppure se iniziate un documento html con qualcosa di diverso dal doctype come un prologo XML, oppure se il doctype non lo mettete proprio. Si è tentato di riassumere le possibili [combinazioni browser/doctype](#) in questa tabella di Henri Sivonen.

Il tag <HTML>

Subito dopo il doctype scriviamo necessariamente il tag <HTML>. Così diciamo agli useragent che quanto è contenuto tra esso e la sua chiusura (</HTML>) è codice html. Possono essere specificati al suo interno anche gli attributi:

- **LANG** (linguaggio) il cui valore è espresso con [sigle standard di due caratteri](#) ["IT" per Italia]
- **DIR** (direzione della lettura) che per i caratteri occidentali è LTR [left to right]
- **XMLNS** (il [namespace](#)) necessario solo per documenti xhtml in quanto derivati da XML

Un esempio:

```
<html xmlns="http://www.w3.org/1999/xhtml" lang="it" dir="ltr">
```

<HEAD>: i META-TAG

Finalmente ci siamo, ora apriamo la nostra sezione <head> e ci mettiamo dentro il primo dei suoi tag: <title> Il titolo della pagina </title> Il suo contenuto fornisce informazioni all'utente comparando nella barra in alto del browser (detta appunto "barra del titolo") ma le fornisce anche ai motori di ricerca essendo uno dei dati che questi prendono in maggior considerazione nell'indicizzare le nostre pagine. Dunque fate molta attenzione nel decidere cosa scriverci. Consiglio: nella homepage usatelo per descrivere il vostro sito, magari inserendoci anche qualcuna delle parole chiave su cui avete puntato. Nelle pagine interne invece potete fornire indicazioni all'utente sul percorso della sua navigazione, es. "Mio sito » Sezione: articoli » CSS". A questo punto vanno inseriti quella serie di metatag la cui sintassi è piuttosto riconoscibile. Eccone alcuni:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<meta name="keywords" content="parole chiave, divise, dalla virgola">
<meta name="description" content="un breve testo descrittivo">
```

Il primo serve a definire il set di caratteri che abbiamo adottato (iso-8859-1 è lo standard per i caratteri occidentali)

Il secondo contiene le parole chiave che volete verranno utilizzate dai motori di ricerca per "trovare" il nostro sito (fatevi furbi: più le parole chiave sono comuni, meno sono efficaci. Lasciate perdere "mp3, sex, free, ecc.ecc.")

Infine "description" ha come contenuto il breve testo (consiglio massimo 300 caratteri) che verrà anch'esso dato in pasto ai motori di ricerca.

Esistono molti altri meta-tag che ora non descriveremo. Servono a gestire i motori di ricerca, fare refresh e reindirizzamenti, fornire informazioni sul sito del tipo: autore, copyright, software usato per generare le pagine, scadenza e aggiornamento dei contenuti. In certi casi, esempio in grandi archivi di documenti, fornire metadati è essenziale per catalogarli, ma è necessario che la forma utilizzata per definirli sia comune a tutti. In questo senso nascono standard di descrizione delle risorse in formato elettronico come il noto [Dublin Core](#). Ora capirete il perchè del "DC" negli esempi qui sotto:

```
<meta name="DC.Creator" content="Mio nome" />
<meta name="DC.Language" content="Italian" />
```

Attenzione però, i 15 elementi descrittivi fornitevi dal DC, devono essere rigorosamente rispettati nella sintassi e nella semantica per essere davvero utili.

Accessibilità e usabilità

Si può rendere un sito accessibile e usabile già in questa parte del codice. Vedremo due esempi in cui ciò avviene.

I. I fogli di stile.

Partiamo dall'assunto che tutte le informazioni necessarie alla formattazione della nostra pagina (la sua "presentazione") le forniremo tramite un foglio .css esterno. Per essere attenti all'usabilità però, i fogli di stile saranno più di uno: avremo quelli legati ai vari device (schermo, stampante, PDA, dispositivi alternativi come braille o sintetizzatori vocali) e quello semplificato necessario per la retrocompatibilità coi vecchi browser (su tutti NN4). Il mio consiglio è di strutturare il codice in questo modo:

```
<link rel="stylesheet" type="text/css" media="screen" href="basic-styles.css"/>
<style type="text/css" media="screen">
<!--
@import url("advanced-styles.css");
-->
</style>
<link rel="stylesheet" type="text/css" media="print" href="print-styles.css"/>
```

In rosa: il link al file *basic-style.css* ovvero al foglio di stile piuttosto semplice che abbiamo creato per i vecchi browser, quelli che non gestiscono molte delle specifiche CSS2.

In verde: poichè i vecchi browser (di quarta generazione) non supportano la regola `@import` ignoreranno questa istruzione e quindi non caricheranno il nostro foglio *advanced-styles.css*, a differenza di quelli moderni.

Infine in giallo: grazie alla definizione `media="print"` forniamo un foglio di stile che verrà usato solo dalle stampanti. [Nota: NN4 non comprende valori diversi da "screen" per l'attributo "media"]

Un appunto: esiste la pratica del "browser sniffing", ovvero tramite javascript si tenta di capire quale browser usa l'utente e gli si passa il foglio di stile adeguato. Io la eviterei, nella maggior parte dei casi, poichè rispetto ai vantaggi che può offrire, essa richiede troppa attenzione sia nella scelta del codice javascript (che potrebbe sbagliarsi o peggio crashare) che nella realizzazione dei diversi fogli.

Naturalmente tutto ciò va valutato sulla base del progetto, caso per caso. Importante è far sì, in ogni modo, che i contenuti siano sempre e comunque accessibili dalla maggior parte degli utenti, seppur non tutti li vedranno esattamente allo stesso modo.

II: I tag link di navigazione

Anche se i browser più diffusi, mi riferisco a IE5.x e IE6, non supportano questa soluzione, dovete sapere che grazie al tag `<link rel="">` si posso creare dei pulsanti standard che appariranno nella barra di navigazione del browser. In questo modo potete fornire velocemente un'alternativa per accedere alle pagine più importanti del vostro sito, come Home, Mappa, Autore, Help, Cerca, capitoli di un documento o fornire tasti come Precedente e Successivo. Degli esempi ed un elenco dei valori possibili dell'attributo REL lo trovate [qui, sul W3C](#).

<LINK rel="alternate" >

Con il tag <LINK> come abbiamo visto possiamo specificare fogli di stile, dare una mano all'accessibilità e, aggiungiamo, linkare altri elementi utili del nostro sito. Negli esempi del W3C [il link citato sopra] c'erano anche altre due possibilità:

```
<link rel="alternate" type="application/rss+xml"
title="RSS" href="tuofeed.rdf" />
<link rel="alternate" href="tuapagina.fr" hreflang="fr"
title="Traduzione francese" />
```

Questo esempio mostra come si possa segnalare alle applicazioni che ne fanno uso (aggregatori, motori di ricerca basati sui feed, ecc.ecc.) la presenza di un feed RSS generato dal nostro sito). Lo stesso si può fare con un altro formato di metadati come FOAF, ma attenzione al giusto formato MIME nell'attributo TYPE (per FOAF è "application/rdf+xml"). Infine, se servisse, si posso fornire versione alternative della pagina in un'altra lingua.

APPROFONDIMENTO:

- SGML è Standard ISO 8879
- HTML 4.01 è l'ultima versione di HTML e risale a Dicembre 1999
- CSS1 nasce nel Dicembre 96, CSS2 nel maggio 98
- Sul W3C: [DTD per HTML4](#) e [XHTML1](#)
- Il quirk sui siti di [IE](#) e [Mozilla](#) e [Opera](#)
- Sulle tecniche di import dei CSS: [Eric Meyer](#)
- L'elenco dei [valori assegnabili a MEDIA](#), dalle specifiche W3C

Christian Fusi | Novembre 2003