



Le nuove tabelle: CSS 3

DARIO DE JUDICIBUS

Usate per anni dagli sviluppatori in modo da supplire alle carenze dell'HTML per quello che riguardava la definizione della struttura di una pagina web, rese poi obsolete in tale ruolo dalle tecniche basate sulle divisioni e i fogli di stile, le tabelle sono state riscoperte dalle ultime versioni dell'XHTML e del CSS riprendendo il loro significato originale.

Abbiamo visto nello scorso numero come l'XHTML 3 abbia ridefinito completamente le tabelle rimuovendo dai vari elementi tutti quegli attributi che si riferiscono al loro aspetto, lasciando solo quelli che ne definiscono la struttura logica e aggiungendo per lo più elementi e attributi utili sul piano semantico (vedi *tabella A* in fondo all'articolo).

Questa separazione dei ruoli, comunque già possibile con l'XHTML 1 e il CSS 2, ora è diventata obbligatoria, non essendo più disponibili negli elementi per le tabelle dell'XHTML 2 attributi come, ad esempio, **width**, **height** e **border**.

Un esempio di divisione rigorosa dei compiti è la tabella riportata in *figura 1*.

Figura 1 - Esempio di tabella

Questa può essere definita, dal punto di vista della struttura, con il seguente blocco di istruzioni XHTML 2

```
<colgroup id="azzurro" span="3" />
<col id="bianco_1" />
<col id="bianco_2" />
<col id="bianco_3" />
<colgroup id="giallo">
<col id="giallo_1" span="2">
<col id="giallo_2" span="2">
</colgroup>
```

mentre dal punto di vista dello stile, richiede le seguenti proprietà:

```
#azzurro { width: 150px; background-color: #BBE0E3 ; }
#bianco_1 { width: 50px }
#bianco_2 { width: 50px }
#bianco_3 { width: 50px }
#giallo { background-color: #FFFFCC }
#giallo_1 { width: 100px }
#giallo_2 { width: 100px }
```

In questo secondo articolo vedremo dunque come si debba usare il CSS 3 per completare il codice XHTML 2 e dare così ad una tabella un aspetto ben definito.

Le tabelle nel CSS 3

Una delle caratteristiche più interessanti delle tabelle, la stessa che ha fatto sì che per diverso tempo queste siano state utilizzate in modo improprio per fornire una struttura alla singola pagina, sono gli algoritmi utilizzati dai browser per bilanciarne la composizione in funzione sia di quanto specificato dal disegnatore in termini di marcatori e attributi, sia del contenuto di ogni singola cella. Mentre le divisioni sono dei meri contenitori e i paragrafi e le liste permettono solo una riorganizzazione elementare del testo, le tabelle hanno la capacità di adattarsi al proprio contenuto in modo da massimizzarne la leggibilità. Da notare che gli algoritmi in questione non sono comunque definiti dagli standard a esclusione del cosiddetto algoritmo a larghezza fissa (**table-layout: fixed**), impostabile appunto nel foglio di stile.

In effetti, gli elementi che compongono una tabella si differenziano così tanto da tutti gli altri elementi da richiedere valori particolari per la proprietà **display** usata nei fogli di stile, ovvero quella proprietà che determina come un elemento debba essere reso in relazione agli altri. Mentre la maggior parte degli elementi, infatti, ricadono sotto la categoria **block** o **inline** o comunque varianti delle stesse, ogni elemento di una tabella ha associata una categoria specifica di visualizzazione. Ne consegue che la proprietà **display** assume i seguenti valori, di norma, per i vari elementi **visualizzabili** di una tabella (escluso quindi **summary**):

```

table      { display: table }
tr         { display: table-row }
thead      { display: table-header-group }
tbody      { display: table-row-group }
tfoot      { display: table-footer-group }
col        { display: table-column }
colgroup   { display: table-column-group }
td, th     { display: table-cell }
caption    { display: table-caption }

```

Senza entrare nel dettaglio dei vari algoritmi, è importante tener presente che in XHTML le tabelle sono dette a *priorità di riga*, ovvero le celle sono suddivisioni delle righe, non delle colonne. Quest'ultime sono una mera conseguenza del posizionamento e dell'ordinamento delle celle nelle varie righe.

Detto questo, sono molte le proprietà caratteristiche degli elementi delle tabelle. Alcune sono le stesse che si usano anche per altri elementi, altre hanno lo stesso nome ma, nel caso delle tabelle, assumono un significato particolare, altre ancora sono specifiche per le tabelle. Facciamo un esempio. Prendiamo l'elemento `col`. Questo elemento può avere quattro proprietà: `border`, `background`, `width` e `visibility`. Fin qui niente di strano: la prima serve a specificare i bordi della colonna, la seconda lo sfondo, la terza la larghezza e la quarta se l'elemento debba essere visualizzato o meno. In realtà le cose sono più complicate di così.

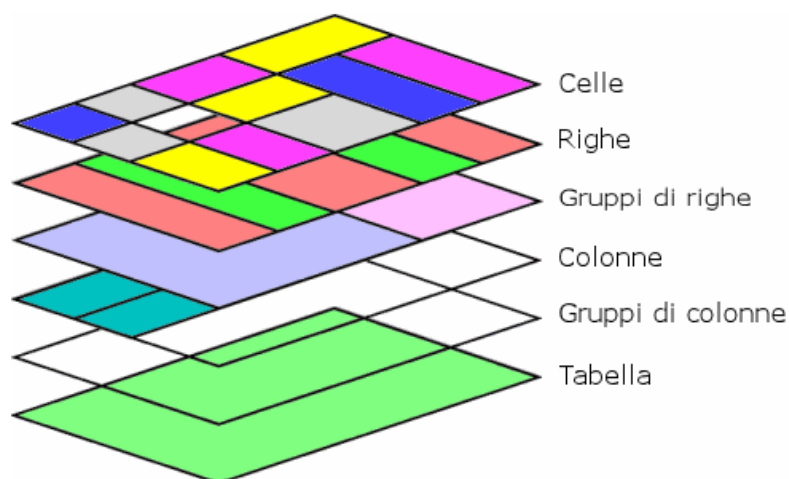


Figura 2 - Sezioni delle tabelle e paginazione

Quando ragioniamo in termini di righe, colonne e celle, dobbiamo immaginare che i vari elementi di una tabella siano posti su diversi piani, come si può vedere in *figura 2 (modello a sei livelli)*. Da notare che mentre le righe e i gruppi di righe ricoprono l'intera tabella, le colonne, i gruppi di colonne e le stesse celle possono anche mancare. Questo in quanto le tabelle in XHTML sono appunto a priorità di riga.

Il livello più basso è quello che contiene la tabella in quanto tale. Al di sopra c'è quello dei gruppi di colonne. Ogni colonna è alta quanto la tabella ma, come già accennato, non è detto che i vari gruppi si sovrappongano alla tabella in modo da coprirla perfettamente. Stessa cosa vale per il terzo livello, quello che contiene singole colonne. Il quarto livello, invece, contiene i gruppi di righe. Ogni riga è larga quanto la tabella ma, in questo caso, i vari gruppi ricoprono completamente la stessa. Analogamente per il quinto livello, che contiene le singole righe. L'ultimo livello, quello più alto, contiene le celle. Ogni riga contiene lo stesso numero di celle, ma dato che alcune possono estendersi per più righe e più colonne e dato che altre possono essere vuote e con lo sfondo trasparente, la struttura finale può essere estremamente articolata.

Tornando alle proprietà, è evidente a questo punto che le caratteristiche di un elemento possono entrare in conflitto con quelle di un altro. Ad esempio, se una cella ha lo sfondo blu ma sta in una colonna gialla, con che colore sarà reso lo sfondo della cella? Se una tabella ha un bordo semplice mentre una riga ce l'ha doppio, nel punto in cui i bordi si sovrappongono, che tipo di linea verrà disegnata?

Per quello che riguarda gli sfondi la risoluzione del conflitto è abbastanza semplice una volta impostato il modello a sei livelli: il colore è quello dell'elemento più in alto, a meno che non manchi o non sia trasparente. In tal caso si va all'elemento immediatamente sottostante e così via fino alla tabella stessa (**table**).

Per i bordi, invece, la questione è un po' più complessa. Esistono due modelli, uno a bordi separati, l'altro a bordi collassati. Quale modello verrà usato dipenderà dal valore della proprietà **border-collapse**. Questa può assumere tre valori: **separate** per il primo modello, **collapse** per il secondo e **inherit**, nel caso di tabelle annidate una dentro l'altra. Il valore iniziale è **collapse**.

Il modello a bordi separati è più indicato quando ogni singola cella ha un proprio bordo. In questo caso la proprietà **border-spacing** serve ad indicare lo spazio tra il bordo di una cella e quello di un'altra, ovvero fa le veci della vecchia proprietà HTML **cellspacing**. Se viene fornito un solo valore, vale sia per la spaziatura orizzontale che per quella verticale; se invece sono forniti due valori, il primo indica la spaziatura orizzontale, il secondo quella verticale. Una seconda proprietà, **empty-cells**, permette di indicare se il bordo va reso anche intorno alle celle vuote (**show**) oppure no (**hide**). Il valore iniziale è **show**. In questo caso non esiste conflitto, dato che il bordo è associato alla cella e non è mai condiviso da due celle.

Nel modello collassato, invece, è possibile specificare un bordo per ogni elemento e persino solo per una parte dello stesso. Ad esempio, si può impostare solo il bordo superiore di una cella, i bordi verticali di un gruppo di colonne, o differenziare i quattro bordi di una o più righe. Ovviamente in questo modello è fondamentale risolvere i conflitti ai quali avevamo accennato in precedenza.

Innanzitutto i bordi sono centrati sulla griglia che suddivide fra loro le celle, per cui metà bordo sta a destra o sopra la linea di suddivisione e l'altra metà sta a sinistra o sotto. Se la larghezza del bordo non è uguale a un numero pari di pixel, il browser dovrà applicare una qualche forma di arrotondamento. Tuttavia, mentre i bordi interni lasciano intoccata la cella, ovvero non si sovrappongono né alla sua larghezza né alla cornice interna o sovrappondo (**padding**), quelli esterni appartengono solo per la metà alla tabella e quindi contribuiscono alla sua larghezza: l'altra metà va a erodere parte dell'eventuale margine intorno alla stessa (fig. 3).

Per quanto riguarda i conflitti, le regole sono le seguenti:

1. se la proprietà **border-style** di un tratto di bordo in uno qualunque dei sei livelli ha il valore **hidden**, allora avrà la precedenza su tutti gli altri, ovvero quel pezzo di bordo non sarà visualizzato;
2. se la proprietà **border-style** di un tratto di bordo in uno qualunque dei sei livelli ha il valore **none**, allora verrà ignorata, ovvero darà la precedenza a tutti gli altri valori di quella stessa proprietà nei vari livelli;
3. un tratto di bordo non sarà visualizzato, quindi se e solo se almeno uno dei valori di **border-style** nei sei livelli è **hidden** o se quella proprietà assume il valore **none** in tutti i livelli;
4. nel caso **border-style** abbia almeno un valore diverso da **none** e non assuma mai il valore **hidden**, allora, a parità di larghezza, hanno la precedenza gli stili più spessi nei confronti dei più sottili, in questo ordine: **solid**, **dashed**, **dotted**, **ridge**, **outset**, **groove** e per ultimo **inset**;
5. nel caso poi i vari bordi abbiano anche larghezze differenti, ha la precedenza quello che ha la larghezza maggiore;
6. infine, se i bordi differiscono solo per il colore, allora ha la precedenza lo stile del livello più alto rispetto a quelli più bassi (ad esempio una cella ha la precedenza su una riga che, a sua volta, ha la precedenza su una colonna).

Un'altra proprietà specifica delle tabelle riguarda la sintesi vocale. Negli ultimi tempi la possibilità di leggere un documento XHTML piuttosto che semplicemente visualizzarlo ha focalizzato sempre più l'attenzione degli standardizzatori. Questa caratteristica non è solo importante per i non vedenti, ma anche per chi utilizza palmari e cellulari e in generale rappresenta un orientamento tecnologico verso un futuro nel quale l'interazione vocale con le macchine sarà sempre più comune. In effetti, da un punto di vista strettamente tecnico, sia l'acquisizione del parlato da parte di una macchina che la sintesi vocale sono tecnologie attuali e ben consolidate. Il problema resta la comprensione da parte di un sistema di quanto detto da un essere umano, ovvero la semantica. Se si limita il tutto a pochi comandi o frasi brevi non c'è già alcun problema, ma se si desidera giungere a un'interazione più sofisticata, allora c'è ancora molto da fare.

Ad ogni modo, leggere una tabella, specialmente se complessa, non è affatto banale. La struttura bidimensionale, spesso non regolare, comporta una serie di problemi. Non è

come leggere un'intestazione o un paragrafo. In definitiva non si può definire una regola generale valida per qualsiasi tabella. Spetta quindi a chi ha disegnato la tabella specificare come questa vada letta. Per questo motivo il CSS 3 mette a disposizione del programmatore la proprietà **speak-header**, che permette di specificare se un'etichetta vada letta una sola volta o ripetuta ogni qual volta viene letto il contenuto di una cella che fa ad essa riferimento.

In definitiva, passare al binomio XHTML 2 + CSS 3 rappresenta un cambiamento sostanziale per quello che riguarda le tabelle. Mentre per gli altri elementi, infatti, si tratta solo di decidere se usare o meno le nuove caratteristiche dei due linguaggi, lo spostamento di tutta una serie di attributi dall'XHTML al CSS richiederà una revisione non indifferente del codice esistente, ovvero una vera e propria migrazione.

TABELLE

Elementi	Attributi	Contenuti
<code>table</code>	comuni	<code>caption?</code> , <code>summary?</code> , (<code>col</code> * <code>colgroup</code> *), ((<code>thead?</code> , <code>tfoot?</code> , <code>tbody</code> +) (<code>tr</code> +))
<code>caption</code>	comuni	(<i>PCDATA</i> testo)*
<code>summary</code>	comuni	(<i>PCDATA</i> flusso)*
<code>col</code>	comuni, <code>span</code>	EMPTY
<code>colgroup</code>	comuni, <code>span</code>	<code>col</code> *
<code>thead</code>	comuni	<code>tr</code> +
<code>tfoot</code>	comuni	<code>tr</code> +
<code>tbody</code>	comuni	<code>tr</code> +
<code>tr</code>	comuni, moduli	(<code>td</code> <code>th</code>)+
<code>td</code>	comuni, <code>abbr</code> , <code>axis</code> , <code>colspan</code> , <code>headers</code> , <code>rowspan</code> , <code>scope</code>	(<i>PCDATA</i> flusso)*
<code>th</code>	comuni, <code>abbr</code> , <code>axis</code> , <code>colspan</code> , <code>headers</code> , <code>rowspan</code> , <code>scope</code>	(<i>PCDATA</i> flusso)*

Tabella A - Elementi, attributi e contenuti delle tabelle