



CSS3 – Una questione di stile

DARIO DE JUDICIBUS

Anche senza utilizzare Java, l'HTML ha permesso fin dalle prime versioni di creare semplici moduli interattivi per l'immissione di dati via Internet. Con il CSS3 questa possibilità si arricchisce da un punto di vista estetico, permettendo così di sviluppare interfacce utente ben integrate nella piattaforma operativa dove gira il browser.

Uno degli aspetti sul quale il CSS3 punta particolarmente è quello dell'interazione con l'utente; in particolare la costruzione di interfacce grafiche basate su HTML e XML e visualizzate grazie appunto ai fogli di stile. Una delle principali novità è il concetto di **apparenza**. Vediamo di cosa si tratta.

L'apparenza

Prendiamo un pulsante nella versione base di Windows XP: in genere è di colore grigio perla, ha un aspetto tridimensionale, rotondeggiante, con una leggera ombreggiatura azzurra che diventa arancione se ci si muove sopra con il cursore; il testo è invece di colore nero, centrato e basato su un carattere senza grazie (*sans-serif*), tipo l'Arial. Ovviamente, se applichiamo uno stile differente al sistema il pulsante cambierà aspetto. D'altra parte lo stesso pulsante in ambiente Mac o Linux ha un aspetto ancora differente e anche qui, a seconda del tipo di interfaccia che si sceglie, avrà determinate caratteristiche.

Ora, se utilizziamo l'HTML o l'XML per costruire una pagina interattiva, ovvero una pagina che permette di visualizzare o introdurre dati e informazioni, ci potremmo trovare a dover fare una scelta: o diamo alla nostra interfaccia un aspetto originale, allineato con lo stile utilizzato nel resto del sito, oppure cerchiamo di farla identica all'interfaccia utilizzata dal sistema operativo su cui gira il browser che visualizzerà il pannello. Nel primo caso dovremo definire ogni singolo aspetto di ogni singolo controllo, nel secondo dovremo semplicemente dire che tipo di controllo intendiamo visualizzare e lasciare al *browser* il compito di acquisire lo stile corrente per quel controllo e applicarlo di conseguenza. In quest'ultimo caso diremo che un certo elemento della nostra pagina ha l'apparenza (*appearance*) di un pulsante (*button*) piuttosto che di una casella di spunta (*check box*).

In realtà il concetto di «apparenza» è legato solo in parte all'elemento. Lo stesso elemento può infatti avere aspetti differenti. Prendiamo una `<select>`. Semanticamente parlando lo scopo di questo elemento è quello di permettere una scelta fra due o più possibilità. Tuttavia tale obiettivo può essere realizzato con una lista semplice (*list box*), a scomparsa (*drop down list*), un menù a tendina (*pull-down menu*) o un gruppo di pulsanti di scelta (*radio button group*). Analogamente una scelta tra due possibilità potrà essere realizzata sia con una casella di spunta che con due pulsanti di scelta. Ecco allora che il foglio di stile non dovrebbe solo permettere di scegliere gli attributi dello specifico controllo, ma anche quale controllo utilizzare per un certo elemento, se esiste la possibilità di poter fare scelte alternative.

Ragionando in termini di interfaccia utente, sono quindi tre le macrocategorie che possono richiedere l'utilizzo dei fogli di stile: l'aspetto, lo stato, l'interazione. Rimangono fuori la semantica, i contenuti, le interdipendenze e la comunicazione. Vediamo perché. La semantica riguarda il significato del modulo che stiamo sviluppando. Ad esempio può essere una scheda riassuntiva di un libro in un catalogo di vendita in rete, oppure una richiesta di supporto tecnico. Tutto si realizza attraverso la realizzazione del documento, ovvero attraverso il linguaggio a marcatori. Per quanto riguarda il contenuto, questo dipende sia dal linguaggio, come nel caso del numero massimo di caratteri che si possono immettere in un campo, sia da un eventuale linguaggio di programmazione come JavaScript, che può ad esempio verificare la validità del dato inserito in un campo. Anche le interdipendenze fra campi vengono solitamente gestite attraverso un qualche linguaggio come JavaScript, Perl, ASP o PHP. Viceversa la trasmissione dei dati dipende sostanzialmente dagli attributi specifici del marcatore `<form>`, come `method` o `enctype`.

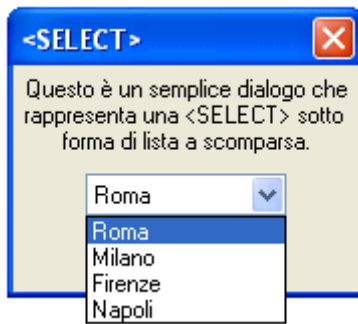
L'aspetto

I fogli di stile, viceversa, hanno la responsabilità di gestire le prime tre macrocategorie. Innanzi tutto l'**aspetto**. Ad esempio, prendiamo il seguente blocco di codice:

```
<select name="prova">
  <option value="1" SELECTED>Roma</option>
  <option value="2">Milano</option>
  <option value="3">Firenze</option>
  <option value="4">Napoli</option>
</select>
```

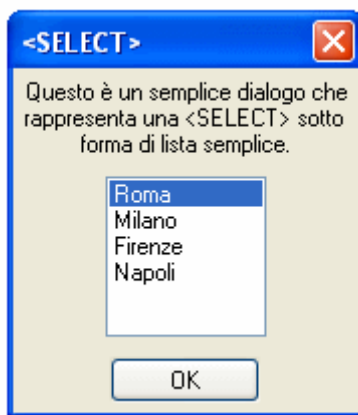
e supponiamo di volerlo rappresentare come una lista a scomparsa, che poi è come in genere i *browsers* rappresentano una `<select>`. Lo stile corrispondente sarà:

```
select#prova {
  appearance: combo-box;
}
```



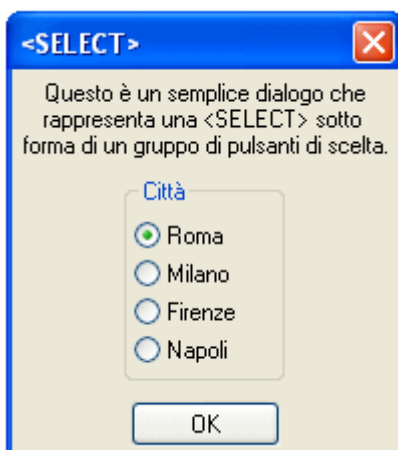
E se invece volessimo una lista semplice, invece che a scomparsa? Semplice: basta modificare lo stile nel modo seguente:

```
select#prova {  
    appearance: list-menu;  
}
```



Terza possibilità: rappresentare le varie scelte come un gruppo di controlli separati, ovvero pulsanti di scelta nel caso possa essere selezionata una sola voce, oppure come caselle di spunta nel caso siano previste scelte multiple. Nel primo caso lo stile sarà:

```
select#prova {  
    appearance: radio-group;  
}
```



Se poi desidero che il controllo in questione non solo sia allineato allo stile del sistema operativo ma che sia anche quello che il sistema utilizza di norma per quel tipo di interazione, mi basterà scrivere

```
select#prova {  
    appearance:normal;  
}
```

che nel caso specifico darà lo stesso risultato di **combo-box**.

Naturalmente il CSS3 ci permette di associare a un certo elemento solo una o alcune caratteristiche di un controllo e di sovrascrivere le altre. Ad esempio

```
input[type=button] {  
    appearance:push-button;  
    color: #fff;  
    background-color: #000;  
}
```



produrrà un pulsante nero con il testo in bianco. Per il resto l'aspetto sarà lo stesso degli altri pulsanti (bordo, eventuale ombreggiatura, comportamento). Per quanto riguarda i colori, sarà possibile specificare, come già in CSS2.1, tutta una serie di nomi che si riferiscono al colore utilizzato dal sistema in quello specifico momento per quella particolare componente del controllo. Ad esempio, **ButtonShadow** è il colore utilizzato per l'ombreggiatura di un pulsante. In CSS3 sono stati aggiunti molti altri colori di sistema rispetto a quelli previsti dal CSS2.1, ma il principio rimane lo stesso. Un discorso analogo riguarda i caratteri. Se ad esempio desidero che i miei pulsanti utilizzino lo stesso tipo di carattere utilizzato dai menù di sistema, mi basterà scrivere

```
input[type=button] {  
    font:menu;  
}
```

Da notare che in questo caso non è stato necessario specificare separatamente la famiglia o le dimensioni del carattere da utilizzare, dato che basta scrivere **menu** e automaticamente sarà utilizzato lo stesso tipo, dimensioni, spessore e caratteristiche di quello usato nelle voci

dei menù. I controlli specificabili come valori per l'attributo **appearance** sono riportati nella tabella A.

In generale, infatti, l'aspetto dell'interfaccia è caratterizzato da tre componenti oltre che dal tipo di controlli utilizzati: immagini, caratteri e colori. Nello specifico **icon**, **font** e **color**. Tutte e tre queste componenti erano già presenti nel CSS2.1, ma con il CSS3 sono state ulteriormente arricchite tanto da permettere oramai lo stesso livello di controllo che si ha con un normale linguaggio di programmazione quando si sviluppa un'interfaccia utente.

Alcuni controlli possono poi essere visti come formati da più componenti. Ad esempio, in una lista a scomparsa abbiamo un campo e una lista, mentre un pulsante di scelta è formato dal pulsante vero e proprio e dal testo posto solitamente alla sua destra. Queste componenti sono identificate in CSS attraverso pseudoelementi. L'unico definito nelle specifiche del CSS2.1 era **::selection**, che corrispondeva alla parte di testo eventualmente selezionata dall'utente con il mouse. A questo il CSS3 ha aggiunto

```
::value  
::choices  
::repeat-item  
::repeat-index
```

Il primo è ovviamente il valore assegnato al controllo, se applicabile, mentre **::choices** rappresenta una lista di possibili scelte. In quest'ultimo caso **::value** rappresenterà la scelta selezionata. Gli altri due pseudoelementi riguardano sequenze di controlli o elementi ripetuti, dove **::repeat-index** rappresenta quello corrente e **::repeat-item** tutti gli altri. Un esempio può essere una griglia di caselle stile foglio elettronico.

Lo stato

La seconda macrocategoria riguarda lo **stato** di un controllo.

Infatti un controllo non ha necessariamente sempre le stesse caratteristiche dato che è un elemento dinamico e quindi cambia aspetto a seconda dello stato in cui si trova. Ad esempio, se desidero evidenziare una casella di spunta quando è selezionata, potrei richiedere che la casella stessa o l'etichetta diventino rosse; oppure potrei modificare il colore del bordo di un pulsante come vi passo sopra il cursore; o ancora rendere quasi invisibile un controllo se non nascondendolo del tutto (**display:none**) se disabilitato. Per far tutto ciò ho bisogno di differenziare i vari stati nel selettore. Un esempio già lo conosciamo ed è la pseudoclasse **:hover**, che permette di definire lo stile di un elemento quando il cursore vi passa sopra. Il principio è quindi lo stesso: ogni stato è associato ad una pseudoclasse e quindi per ogni stato posso stabilire uno stile differente. Ad esempio, gli stili

```
input[type=button] {  
  border:4px outset;  
}
```

```
input[type=button]:hover {  
    border:4px inset;  
}
```

fan sì che come il cursore passa su un pulsante questo rientri apparentemente nel piano, come se qualcuno lo avesse schiacciato.

Per rappresentare gli stati di un controllo quindi, o meglio, per indicare in un foglio di stile quali caratteristiche deve avere un controllo quando è in un certo stato, il CSS usa le pseudoclassi. Come già detto, alcune erano già state definite nel CSS2.1, ovvero

```
:hover  
:active  
:focus  
:enabled  
:disabled  
:checked  
:indeterminate
```

Il CSS3 aggiunge a queste anche

```
:default  
:valid  
:invalid  
:in-range  
:out-of-range  
:required  
:optional  
:read-only  
:read-write
```

il cui significato è abbastanza evidente dal nome. Le varie pseudoclassi possono anche essere applicate contemporaneamente, là dove ha un senso. Ad esempio, una casella di spunta può essere allo stesso tempo disabilitata e smarcata, ovvero

```
input[type=checkbox]#country:disabled:checked {  
    color: #808080;  
}
```

così come un pulsante può essere allo stesso tempo quello di *default* ed avere il fuoco:

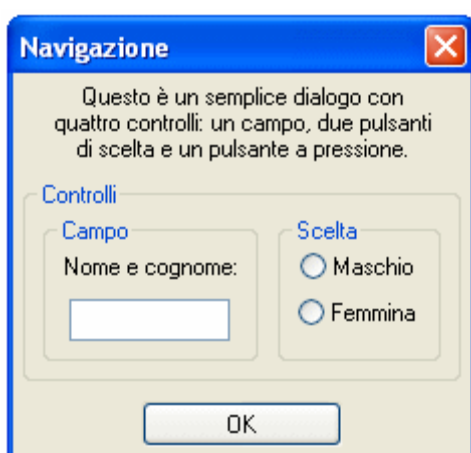
```
input[type=button]#submit:default:focus {  
    color: #808080;  
}
```

Queste pseudoclassi permettono quindi di alterare dinamicamente lo stile di un controllo a seconda dello stato. Si può ad esempio evidenziare un campo nel quale è stato immesso un valore non valido. Attenzione però: il CSS non si occupa di verificare la validità del valore all'interno del campo. Come già detto, tutte le questioni semantiche e sintattiche devono essere prese in carico da qualcun altro. Tuttavia, se il linguaggio XML prevede un meccanismo di convalida semantica, allora il CSS si occuperà di attivare lo stile giusto a seconda dello stato che verrà assegnato dall'XML al controllo in questione.

L'interazione

E veniamo alla terza e ultima macrocategoria. L'**interazione** fra utente e interfaccia avviene generalmente attraverso due meccanismi: tastiera e puntatore (*mouse*). Naturalmente ne esistono altri, come i vari controlli per i giochi (*joystick, game pad, game wheel*) e la stessa voce, almeno per i sistemi dotati di comandi vocali, ma possono comunque sempre essere ricondotti ai primi due, almeno per quanto riguarda le interfacce di tipo tradizionale, ovvero quelle non legate ai giochi.

Un elemento già presente nella CSS2.1 era il cursore di puntamento (**cursor**) che nella CSS3 si è arricchito di molti altri valori. Assolutamente una novità invece le proprietà relative alla navigazione all'interno di un pannello di controlli, navigazione che in genere si effettua via tastiera premendo il tasto di tabulazione. La prima è **nav-index**, che permette di specificare l'ordine nel quale i vari controlli devono essere selezionati premendo **TAB**. Altre quattro proprietà permettono di specificare su quale controllo il fuoco deve andare quando viene premuta una delle quattro frecce di direzione: su, giù, destra e sinistra. Facciamo un esempio. Supponiamo di avere un'interfaccia con quattro controlli: un campo, due pulsanti di scelta e uno a pressione. Il campo deve essere il primo controllo. Se premo il pulsante di tabulazione più volte, il fuoco dovrà quindi passare dal campo al pulsante **OK** passando per i due pulsanti di scelta, prima **Maschio**, poi **Femmina**.



Vediamo come fare:

```
input[type=text]#name {  
    nav-index: 1;  
    nav-right: #male;  
    nav-left: #male;
```

```

    nav-up: #ok;
    nav-down: #ok;
}
input[type=radio]#male {
    nav-index: 2;
    nav-right: #female;
    nav-left: #name;
    nav-up: #name;
    nav-down: #female;
}
input[type=radio]#female {
    nav-index: 3;
    nav-right: #ok;
    nav-left: #name;
    nav-up: #male;
    nav-down: #ok;
}
input[type=button]#ok {
    nav-index: 4;
    nav-right: #name;
    nav-left: #female;
    nav-up: #female;
    nav-down: #name;
}

```

Come si può vedere, oltre alla sequenza di navigazione sono stati indicati quali controlli debbano essere selezionati usando le frecce di direzione. L'obiettivo è quello di cercare di rendere la navigazione il più semplice e intuitiva possibile.

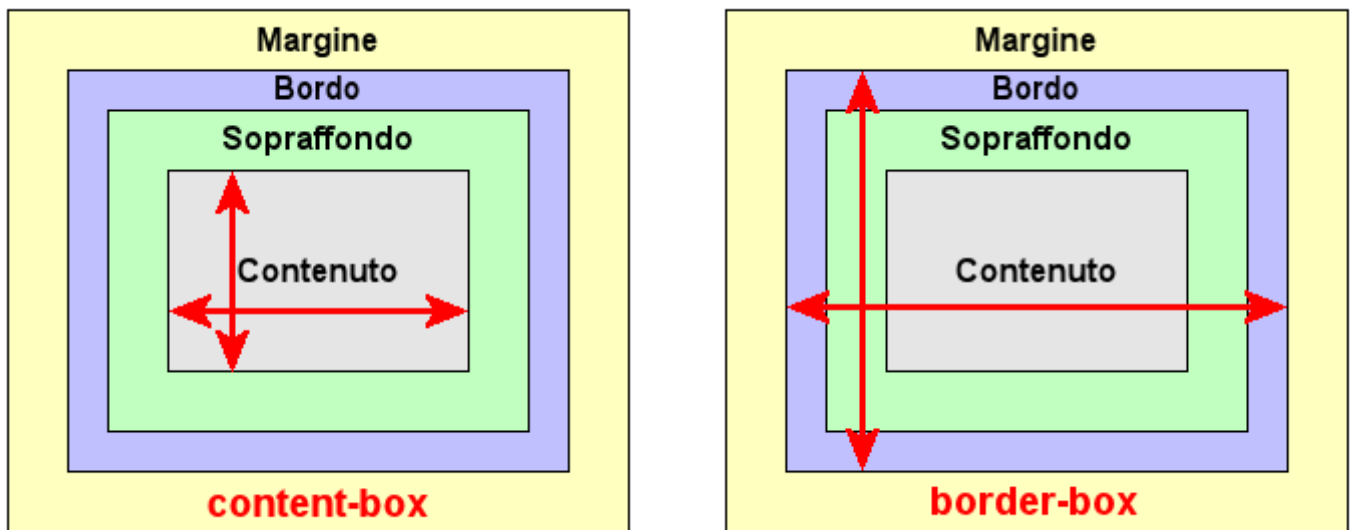
Posizionamento e ridimensionamento

Per concludere vediamo le ultime tre proprietà del modulo CSS3 base per le interfacce interattive. Abbiamo scorporato queste proprietà dalle altre in quanto, pur riguardando le prime due l'aspetto e la terza l'interazione, si possono applicare estensivamente anche a tutti gli altri elementi del linguaggio, non solo a quelli tipici dei moduli interattivi (**form**).

La prima riguarda il cosiddetto «modello a scatola» (*box model*), disperazione di tutti gli sviluppatori che devono creare siti che siano identici sia in *Internet Explorer* che in *Mozilla* o in altri *browser*. Il modello a scatola è così denominato perché indica come è fatta appunto la scatola (*box*) che contiene un elemento.

Le specifiche originali del CSS prevedevano che le dimensioni di un elemento non includessero né il margine (**margin**), né il bordo (**border**), né il sovrappiù (**padding**) della scatola. Purtroppo la Microsoft ha interpretato male quelle specifiche per cui tutte le versioni precedenti alla 6 di IE hanno incluso nelle dimensioni anche bordo e sovrappiù. Questo ha costretto i programmatori a sviluppare tutta una serie di trucchetti (*hack*) allo scopo di dare alle pagine lo stesso aspetto in tutti i *browser*.

In CSS3 è possibile specificare per ogni elemento il modello a scatola da applicare: quello originale (**content-box**) o quello “male interpretato” (**border-box**).



In aggiunta al bordo, è poi stato definita una nuova proprietà decorativa denominato *contorno* (**outline**). Il contorno non fa parte dell'elemento al quale si applica e quindi non altera il modello né le dimensioni. Non solo: un contorno non deve necessariamente essere rettangolare ed è sempre disegnato sopra la scatola che contiene l'elemento. Ad esempio, se un testo è spezzato in varie righe, il contorno segue la forma del testo, anche se non è rettangolare.

In aggiunta al bordo, è poi stato definita una nuova proprietà decorativa denominato *contorno* (**outline**). Il contorno non fa parte dell'elemento al quale si applica e quindi non altera il modello né le dimensioni. Non solo: un contorno non deve necessariamente neanche essere rettangolare ed è sempre disegnato sopra la scatola che contiene l'elemento. Ad esempio, se un testo è spezzato in varie righe, il contorno segue la forma del testo, anche se non è rettangolare.

Per il resto il contorno si definisce come si fa con il bordo, ovvero fornendo uno spessore, uno stile e un colore. Inoltre è possibile indicare a che distanza deve essere tracciato rispetto al bordo (*offset*). Oltre che a evidenziare del testo, il contorno può servire, ad esempio, a segnalare quale controllo abbia il fuoco in un pannello:

```
:focus, :active {  
    outline: thin dotted silver ;  
    outline-offset: 2px ;  
}
```

Infine la proprietà di ridimensionamento. Questa proprietà permette di indicare se un certo elemento è ridimensionabile da parte dell'utente e, nel caso, lungo quale direzione. Ad esempio

```
iframe.image {
```

```
resize: both ;  
overflow: auto ;  
}
```

permette di ridimensionare tutti gli elementi `<iframe>` di classe `image` in entrambe le dimensioni.

Questo è quanto per quello che riguarda il CSS3 e lo sviluppo di moduli interattivi. Ovviamente tutte queste proprietà sono soggette a cambiamenti e aggiunte, dato che le specifiche CSS3 sono ancora in fase di elaborazione. Ad esempio, proprio recentemente è stata eliminata la proprietà `key-equivalent`, presente nella versione precedente delle specifiche CSS3. Tuttavia questo modulo è fra quelli già proposti per diventare *raccomandazione* (*Candidate Recommendation*), per cui è già abbastanza stabile.

TABELLA A

icon		Una piccola immagine, detta “icona” in italiano, che rappresenta un oggetto, spesso associata a un nome o ad un’etichetta.
window		Un’area dotata di un bordo, detta “finestra” in italiano, che serve a contenere altri elementi per permettere l’interazione con l’utente.
	desktop	Una finestra che rappresenta il sistema nel suo complesso e che generalmente contiene altre finestre.
	workspace	Una finestra che rappresenta un progetto o un’applicazione, generalmente caratterizzata da un’intestazione.
	document	Una finestra che rappresenta un documento, generalmente dotato di intestazione. Può anche essere utilizzato per rappresentare cartelle contenenti altri documenti.
	tooltip	Una finestrella utilizzata per mostrare temporaneamente alcune informazioni di ausilio all’utente. In CSS2 veniva chiamata info .
	dialog	Una finestra che permette all’utente di eseguire una specifica azione come effettuare una scelta o introdurre dati. Può anche essere utilizzata per fornire dati all’utente o mostrare un messaggio. Chiamata message-box nel CSS2.
button		Un piccolo oggetto, detto “pulsante” in italiano, che permette all’utente di fare una scelta. In genere è fornito di un’etichetta.
	push-button	Un pulsante dotato di bordo, spesso con un’apparenza tridimensionale, che può essere premuto per attivare una specifica azione. In CSS2 era chiamato caption .
	hyperlink	Un pulsante che rappresenta un collegamento ipertestuale, generalmente rappresentato come un testo sottolineato.
	radio-button	Un pulsante, detto “di scelta” in italiano, rappresentato come un cerchietto accanto ad un’etichetta di testo. Se premuto, il cerchietto diventa nero e il pulsante si dice smarcato. A volte può trovarsi in uno stato indeterminato (né smarcato, né non smarcato).
	checkbox	Analogo al radio-button , la “casella di spunta” è spesso rappresentata come un quadratino al cui interno compare, se premuto, una crocetta o un segno di spunta. Anch’esso può esistere in uno stato indeterminato.
	menu-item	La “voce” all’interno di un menù (vedi dopo). Pur facendo parte di un menù, una voce è considerata equivalente a un pulsante a tutti gli effetti. A volte una voce rappresenta il punto di apertura di un sottomenù.
	tab	Un pulsante che rappresenta la linguetta di un pannello in un’interfaccia a libretto.
menu		Un insieme di opzioni tra le quali l’utente può scegliere, detto “menù” in italiano. La scelta può essere singola, multipla o alternata.
	menubar	Un menù di menù, generalmente organizzati sotto forma di barra orizzontale. A volte, tuttavia, può essere verticale, posto lateralmente nel documento.
	pull-down-menu	Un menù il cui nome è visibile ma le cui scelte non lo sono fintanto che non si preme sul nome. In tal caso, generalmente, si apre una tendina con le varie scelte, tendina che poi scompare a scelta avvenuta.
	pop-up-menu	Un menù nel quale tutte le opzioni tranne quella al momento attiva sono invisibile fintanto che l’utente non attiva il menù. Quando questo succede,

	generalmente si apre un riquadro che permette di cambiare la scelta, dopodiché il riquadro torna ad essere invisibile.
list-menu	Una “lista” di opzioni fra le quali l’utente può selezionare una o più scelte. È sempre visibile.
radio-group	Un menù dove le opzioni sono rappresentate come pulsanti di scelta.
checkbox-group	Un menù dove le opzioni sono rappresentate come caselle di spunta.
outline-tree	Un menù dove le opzioni possono essere mostrate o nascoste a seconda dello stato di un piccolo pulsante, generalmente di forma triangolare.
range	Un controllo che permette di scegliere un’opzione all’interno di un intervallo. In genere è rappresentato come una barra nella quale un cursore scorrevole indica il valore attualmente selezionato. A volte è invece rappresentato come una manopola che può ruotare. In questo caso il valore corrente è rappresentato da un segno sulla manopola.
field	Un’area, detta “campo” in italiano, che permette all’utente di immettere o modificare un valore, generalmente utilizzando la tastiera. A volte un campo non è modificabile ma è usato solo per mostrare il valore.
combo-box	Un campo associato a un menù di valori predefiniti che possono essere utilizzati al posto dell’immissione diretta da parte dell’utente. A volte l’utente può introdurre nel campo anche altri valori, altre volte è obbligato a selezionare uno di quelli predefiniti.
signature	Un campo per l’immissione di una firma.
password	Un campo per l’immissione di una parola d’ordine la quale, in genere, non viene visualizzata mentre viene immessa. Solitamente, man mano che l’utente introduce nuovi caratteri, nel campo appare un pallino o un quadratino nero.