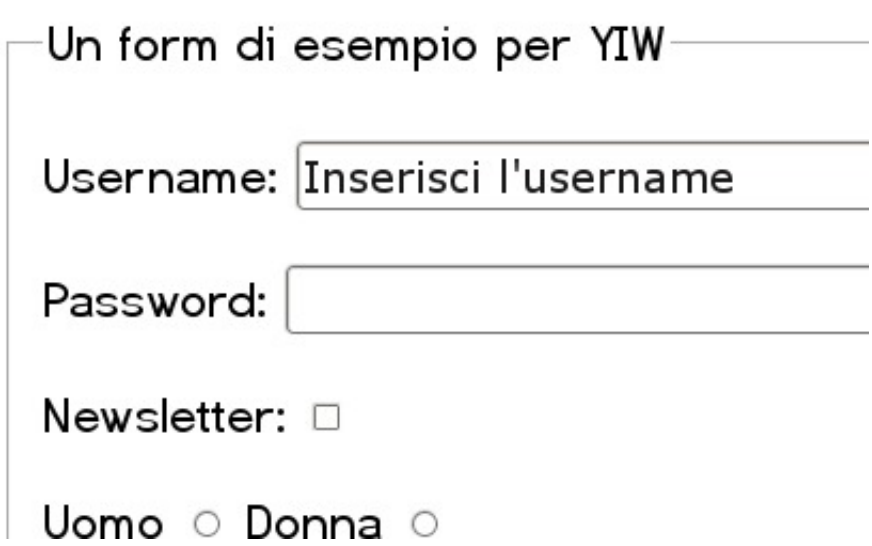
Web design elements: i form (le basi)



I form sono l'anima di Internet. Senza questi elementi, il Web così come lo conosciamo oggi non esisterebbe. Pensa ai siti di ecommerce,ai pannelli di amministrazione, social network, blog (e relativi commenti), senza form. Ma, come capita spesso, le cose più importanti, sono spesso relegate ad un aspetto di secondo piano. E così spesso la struttura, il design visuale e tutte le questioni legate ai form vengono realizzate in poco tempo, alla fine del progettazione del sito. Con questa serie di articoli studieremo più a fondo le potenzialità e i limiti di questi elementi: questa settimana parleremo dei vari componenti di un form.

Qual sono gli elementi che compongono un form?

L'elemento principale per definire i form in (X)HTML è chiaramente il tag form:

Come puoi vedere questo tag prevede due attributi: action e method. Il primo, obbligatorio, definisce *dove* il browser deve inviare i dati del form, una volta che è stato cliccato il pulsante di invio; il valore deve essere l'indirizzo di uno script che sa come gestire i dati.

Il secondo attributo, invece, definisce *come* tali dati devono essere inviati. I metodi fondamentali sono due: get e post. Il metodo get (che è quello predefinito) invia i dati in chiaro allo script, e li accoda all'url: questo è il metodo, ad esempio, con cui vengono inviati i dati tramite il form di Google. Il vantaggio è che gli utenti possono salvare questi URL e riutilizzarli in un secondo momento.

Con il metodo post i dati vengono sempre inviati **in chiaro** attraverso gli header HTTP della pagina ma non essendo visibili ad occhio nudo è più complicato recuperarli. Utilizzare il metodo post è il

primo passo da intraprendere per la sicurezza di un form. Ad esempio i campi di immissioni per le carte di credito sono inviati sempre tramite questo metodo.

L'attributo name

Un concetto importante per i form è **il riconoscimento dei dati**. Infatti lo script destinatario del form non ha alcun modo distinguere ad esempio una password da un nome utente. È importante, quindi, definire cosa rappresentano i dati attraverso l'attributo name, in modo tali da poterli gestire senza difficoltà.

Il tag input

La maggior parte degli elementi di un form viene resa attraverso il tag input. Infatti a seconda del valore dell'attributo type verrà mostrato un controllo differente. I valori accettati da questo attributo sono dieci:

1. text
2. password
3. checkbox
4. radio
5. submit
6. reset
7. file
8. image
9. hidden
10. button

text e password

Questo elemento viene reso come un campo di inserimento su una riga, in cui l'utente può inserire il testo.

È possibile inserire un attributo maxlength per stabilire il numero massimo di caratteri che è possibile immettere, in questo modo:

Inoltre, è possibile stabilire se deve essere presente un testo predefinito al caricamento della pagina, tramite l'attributo value:

Il campo di tipo password è analogo a quello text, con l'unica differenza che il testo viene sostituito da asterischi (o altri elementi grafici, a seconda del browser). L'idea alla base di questo approccio è

che un eventuale utente che "sbirci" il monitor non possa leggere ciò che viene digitato.

Puoi notare che ho aggiunto a quest'ultimo campo l'attributo size che stabilisce quanti caratteri deve essere lungo il campo.

checkbox e radio

In alcune occasioni, gli utenti devono essere obbligati a scegliere tra alternative predeterminate. I checkbox permettono di scegliere tra più voci quelle che interessano. L'HTML è:

Possiamo scegliere se, inizialmente, la casella deve essere vistata, aggiungendo l'attributo checked:

Il valore che verrà inviato al server sarà quello dell'attributo name, altrimenti quello dell'attributo value, se presente. Ecco un esempio.

I bottoni radio sono concettualmente simili ai checkbox, con la differenza che permettono di scegliere un solo valore tra quelli disponibili. Per raggruppare più bottoni radio si utilizza l'attributo name, e quando uno viene selezionato, automaticamente gli altri saranno deselezionati.

È importante inserire, in questo caso, l'attributo value; in caso contrario qualsiasi bottone radio sia selezionato, verrà inviato allo script solo il valore "name=on" (nell'esempio, "gender=on"). Esempio

submit e reset

Probabilmente uno degli elementi più importanti, gli input di tipo submit, servono ad inviare tutti i dati immessi nel form al server. Il valore visualizzato sul pulsante viene definito dall'attributo value, altrimenti è quello predefinito dal browser.

Il campo reset, invece, serve per "azzerare" tutti gli input inseriti, riportandoli a quelli predefiniti. È utile in caso di errori multipli commessi dall'utente, anche se sta via via scomparendo dalle pagine web di oggi.

file

Il campo file permette all'utente di scegliere un file da caricare sul server. Al momento dell'invio del

form, anche il file verrà inviato insieme agli altri dati. Per far sì che funzioni, però, il metodo di invio deve essere post e bisogna aggiungere l'attributo enctype al tag form, in questo modo:

Esempio

image

Il tipo di input image oggi è utilizzato spesso come alternativa grafica al tipo submit. Infatti, permette di selezionare un'immagine che, cliccata, invia il form (nella "preistoria" del web veniva utilizzato anche per realizzare mappe di immagini). Il markup è semplice:

Esempio

hidden e button

Il tipo hidden crea un campo nascosto all'utente, che quindi non può sapere della sua esistenza (a meno di guardare nel codice). È utlizzato spesso, in combinazione con script lato server (tipo PHP, per intenderci) ad esempio per assicurare dell'integrità dei dati inviati. Il suo utilizzo è semplice:

Il campo button, invece, non fa assolutamente nulla (almeno per quanto riguarda i dati del form): può essere utilizzato insieme a degli script javascript per attivare degli effetti decisi dallo sviluppatore. Puoi vedere degli esempi nella documentazione di jQuery, dove vari elementi button vengono utilizzati per mostrare gli effetti delle animazioni.

textarea e select

Esistono due campi per i form che non si possono ottenere dal tag input. Il primo è textarea che permette agli utenti di inserire qualunque tipo di testo, anche su più righe. L'utilizzo è leggermente diverso da input: infatti ha sia un tag di apertura che uno di chiusura, ed il valore iniziale è compreso fra questi due. Ad esempio:

Nota gli attributi rows e cols che definiscono la larghezza e l'altezza dell'area di testo. Questi due attributi (che sostanzialmente sono presentazionali) sono obbligatori, anche se possono essere ridefiniti attraverso CSS.

L'altro elemento è rappresentato dai campi select. Questo elemento permette di scegliere da un menu delle opzioni. È utile quando si hanno molte opzioni tra cui far scegliere gli utenti, e utilizzare campi radio o checkbox affollerebbe troppo la pagina.

Come textarea ha un tag di apertura e uno di chiusura, e racchiude al suo interno una o più option:

Esempio

È possibile visualizzare un valore e inviarne un altro utilizzando l'attributo value (ad esempio, selezionando Fabrizio de Andrè nell'esempio verrà inviata la stringa "fda"); inoltre è possibile decidere quale elemento visualizzare al caricamento della pagina, con l'attributo "selected" (nell'esempio "Bob Dylan").

Di default è possibile visualizzare un solo elemento per volta, selezionando quello interessato da un menu a tendina. È possibile fissare quanti elementi mostrare tramite l'attributo size:

Esempio

Spesso in questo caso, si vuole permettere all'utente di scegliere più di una opzione, in questi casi interviene l'attributo multiple:

Se poi gli elementi sono molti, è possibile organizzarli in sottogruppi, in modo da facilitare la selezione all'utente. Per far questo, si utilizza il tag optgroup:

Esempio

Accessibilità: fieldset, legend e label

Dato che i form sono una parte fondamentale per l'interattività di Internet, **è importante che siano accessibili e utilizzabili da tutti gli utenti**, anche quelli limitati da impedimenti fisici. Qualcuno potrà pensare che è un "di più", ma si sbaglia. È giusto che noi web designers progettiamo per il pubblico più ampio possibile, non lasciando che le disparità già presenti nella vita reale vengano trasportate nel web, dove tutti possiamo essere veramente uguali.

I tag che si occupano di dare significato ai vari campi dei form sono fieldset e legend e label. I primi servono per suddividere il form in aree semantiche diverse: ad esempio, possiamo suddividere un form di registrazione in informazioni personali e informazioni di domicilio:

Il tag label fornisce un'etichetta per ogni elemento del form, permettendo di legarli a livello di codice. Facciamo subito un esempio:

Come puoi vedere, nel primo caso, il campo "Username" è descritto attraverso del testo libero, mentre nel caso di "Email" attraverso l'attributo for, l'etichetta è legata all'ID dell'input sottostante.

Visualizzando la pagina in un browser, **la resa è sostanzialmente identica**: la differenza è utenti ipovedenti che utilizzino screen reader potranno essere coscienti che l'etichetta "email" si riferisce a quel determinato campo, cosa non possibile per il campo "username".

Nota che per realizzare il collegamento, ho dovuto inserire l'ID: infatti l'attributo name serve a dare un significato ai dati inviati dal form, l'ID invece viene utilizzato per modificare gli stili (tramite CSS) e per l'elemento label.

Per aumentare l'accessibilità dei nostri form possiamo ancora andare oltre, e specificare l'ordine con cui vengono selezionati i vari elementi tramite tasto TAB, attraverso l'attributo tabindex.

Esempio

In conclusione, ti mostro un form (fittizio) con tutti gli elementi di cui abbiamo parlato:

Conclusioni

Questa è stata la prima lezione alla scoperta dei form. Abbiamo chiarito le varie differenze che ci sono tra i diversi campi di input. La prossima settimana vedremo quali sono le alternative per strutturare un form e come utilizzare i CSS per modificarne l'aspetto.

- I form: elementi di base
- I form: struttura e aspetto
- I form: jQuery e considerazioni
- Come realizzare un modulo contatti in PHP