Quali sono i principali metodi per integrare JavaScript nella tua pagina html?

Negli ultimi anni, con l'introduzione di JavaScript le nostre pagine web sono diventate sicuramente più "vive": effetti fantastici, caricamenti asincroni, interfacce migliorate, hanno reso l'esperienza utente molto più coinvolgente.

All'esponenziale crescita di JavaScript è corrisposto un crescente utilizzo dei link non più come collegamenti ipertestuali, bensì come "pulsanti di azione". In altre parole, **sono dei link vuoti, che servono solo ad innescare un'azione JavaScript**. Sebbene tale pratica non sia vista di buon occhio dai puristi del codice (un link che non porta a nulla, non è un link per definizione), non si può negare che sia molto diffusa. Oggi vedremo alcuni metodi per implementare tali link, e i relativi pro e contro.

Primo metodo: inserire il codice javascript nell'attributo href

Probabilmente il metodo più semplice è quello di **inserire la nostra funzione JavaScript direttamente all'interno dell'attributo href del link**, segnalando al browser che si tratta di un comando JavaScript, in questo modo:

Come puoi vedere nell'esempio 1, il link funziona correttamente in caso di JavaScript abilitato; in caso contrario invece no, rendendo inusabile il link.

Un piccolo miglioramento si può ottenere **utilizzando l'attributo "onclick"** per il link in questione, inserendo un valore fittizio (#) all'interno dell'attributo "href", in questo modo:

Parlavo di piccolo miglioramento, perché in caso di JavaScript disabilitato, il browser non si trova gestire l'evenienza di un attributo "href" vuoto (come nel caso precedente). Ma in realtà **questo metodo è sconsigliato** perché, come puoi vedere cliccando sul link d'esempio 2, una volta eseguito il codice JavaScript, il browser interpreta il cancelletto (#) come un link all'inizio della pagina, scrollando in alto: un effetto decisamente spiacevole.

Si può ovviare a questo inconveniente inserendo una "funzione vuota" all'interno dell'attributo "href" e il nostro codice da eseguire nell'attributo "onclick" così:

Questo metodo risolve il problema dello scrolling in alto però, oltre a ripresentare la possibilità di attributo "href" vuoto, **è noto per causare alcune incompatibilità imprevedibili in determinate situazioni**: la non prevedibilità rende trovare questo tipo di errori un'attività molto difficile e

frustrante: bisogna quindi usare questa tecnica con molta cautela.

Qual è il metodo migliore?

Dopo questa breve panoramica, ti starai chiedendo quale sia il metodo migliore per rendere i tuoi link più accessibili. Il difetto di tutte le metodologie esposte precedentemente è sostanzialmente uno: **non prevedono nessuna alternativa per chi naviga con JavaScript disabilitato** e, al contrario di quanto si possa pensare, questa percentuale non è affatto insignificante. Ti basti pensare al sempre crescente numero di persone che navigano con dispositivi mobile, senza dimenticare chi ha lo scripting disabilitato per ragioni di sicurezza, o chi naviga utilizzando screen-readers o browser testuali.

Riprendendo il discorso, il metodo più semplice per migliorare gli esempi precedenti è **dare un'alternativa all'azione JavaScript**, magari preparando una versione del sito in linguaggio server-side (PHP, C#, Ruby) oppure più banalmente preparando una pagina che informa l'utente che il sito funziona solo con lo scripting abilitato. Il link potrebbe quindi diventare:

Vediamo questo metodo in azione nell'esempio 3: se hai cliccato sul link avrai notato che è stato mostrato il testo nella finestra informativa (quindi è stata avviata l'azione JavaScript), e poi il browser si è portato alla pagina "senzaJs.html". Il motivo di questo comportamento è che all'atto del click su un link il browser controlla se è presente l'attributo onclick: in caso affermativo esegue il codice, ma "ricorda" che l'azione principale che deve eseguire è quella di seguire il collegamento, cosa che avviene non appena la funzione JavaScript termina la sua esecuzione.

Idealmente, quindi servirebbe un modo per "avvisare" il codice JavaScript: "se vieni eseguito, escludi l'azione principale del link". Questo effetto si ottiene utilizzando "return false;" come valore di ritorno del nostro codice JavaScript, in questo modo:

Questa tecnica è illustrata nell'esempio 4: abbiamo finalmente raggiunto il nostro scopo!

Possiamo ancora migliorare però: avrai sicuramente notato che, dopo aver cliccato il link, è rimasto visibile il bordo (l'outline). Tale bordo, oltre ad essere antiestetico, è anche fuorviante, poiché evidenzia una parte della pagina che ha cessato la sua azione: procediamo quindi ad eliminarlo. In prima battuta si potrebbe pensare di disattivare l'outline da tutti i link tramite CSS, con una regola del genere:

ma ciò renderebbe difficile seguire i nostri link utilizzando il tasto TAB. Dato che il problema si presenta solo quando JavaScript è attivo (infatti se è disattivato, l'outline non è visibile, in quanto il browser dirigerebbe l'utente alla pagina "senzaJS.html"), **utilizziamo JavaScript per eliminare**

l'outline. In pratica simuliamo un click all'esterno dell'area del link, in modo da deselezionarlo. Per far ciò utilizziamo la funzione "blur()":

Ho utilizzato la parola chiave "this" che contiene sempre un riferimento all'oggetto che chiama la funzione: a questo viene applicata la funzione "blur()", il cui compito non è altro che deselezionare un oggetto, mimando il clic del mouse all'esterno della sua area. Come puoi vedere nell'esempio 5, il link esegue correttamente il codice JavaScript e viene deselezionato, nascondendo così l'outline.

Con questa tecnica i nostri link sono finalmente accessibili, esteticamente piacevoli e funzionali. **Ma possiamo ancora andare oltre?**

Un passo avanti: eliminare i riferimenti JavaScript dal markup

Gli esempi mostrati finora si riferiscono tutti a codice JavaScript *inline* ovvero impiantato direttamente nel markup HTML. Anche se questa metodologia è più immediata da comprendere, è bene applicare il principio di separazione spostando tutto il codice JavaScript in file esterni, per almeno due buoni motivi:

- **Manutenibilità**: dato che il codice JavaScript è sparso su varie pagine HTML, in caso di errori il debugging è un vero e proprio incubo. I file esterni invece circoscrivono l'area di interesse, facilitando il compito.
- **Caching**: ogni volta che un utente richiede una pagina, oltre al markup deve essere scaricato anche il codice Javascript, con tempi di caricamento mediamente più lunghi. Grazie al caching (se abilitato) i file JavaScript esterni vengono scaricati un'unica volta.

Detto questo, vediamo uno dei metodi per portare le funzionalità viste finora "all'esterno". Possiamo percorrere due strade: JavaScript puro o tramite un Framework JavaScript.

Se non utilizziamo già un framework (come jQuery) all'interno del nostro sito, inserirlo solo per migliorare i link può sembrare **uno spreco di risorse inutile**: quindi si può pensare di risolvere utilizzando una funzione JavaScript scritta ad hoc. Il markup di cui abbiamo bisogno è il seguente:

Tutto qui: basta specificare l'attributo "href" per la versione HTML e un ID, che servirà da ancora per il codice JavaScript.

Metodo 1: JavaScript "liscio"

Apriamo dunque un nuovo file e definiamo la nostra funzione:

Il primo passo è quello di trovare il link interessato nella pagina. A questo scopo, utilizzeremo la funzione "getElementById();":

Il secondo passo è quello di applicare a questo link una funzione con il codice da eseguire, seguito dalla chiamata a "blur()" e a "return false":

Ciò che resta da fare è associare questa funzione al caricamento della pagina, tramite la funzione "onload":

Puoi vedere il funzionamento di questo metodo nell'esempio 6, che mostra un semplice messaggio a video.

Metodo 2: utilizzando un framework

Se già utilizzi un framework per il tuo sito, integrare queste funzioni è ugualmente semplice *(Nota: l'esempio si riferisce a jQuery)*: come nel caso precedente, dobbiamo prima selezionare i link e poi associare all'evento click la nostra funzione. Innanzitutto, creiamo un link con un ID diverso dal precedente:

Il codice JavaScript deve essere caricato non appena l'albero DOM della pagina è stato caricato, quindi utilizziamo la funzione "ready()":

Utilizzando la funzione $ (alias per jQuery), selezioniamo il nostro link:

Infine associamo il nostro codice all'evento click, utilizzando la funzione omonima:

Questo è quanto: possiamo vedere questo metodo utilizzato nell'esempio 7.

Conclusioni e considerazioni

JavaScript è un linguaggio semplice da imparare ed usare, ma questo non significa che dietro l'implementazione di un effetto grafico o di una funzionalità non debba esserci un'attenta pianificazione. Sviluppando per il web, tutti gli elementi devono lavorare all'unisono per consentire la miglior esperienza possibile alla più ampia platea possibile.