AJAX: impariamo a utilizzarlo

Sono appena passate le feste e non ne possiamo più di panettone e canzoni natalizie e luci colorate per le strade… Forse non è proprio così, rimpiangiamo quel clima di festa, ma non tarderà a tornare. Intanto, è tempo di ricominciare e di aggiungere nuove competenze al nostro curriculum.

Cos'è AJAX

Se hai provato anche a dedicarti al lato server-side, o magari collabori con qualche web developer, sicuramente conoscerai il linguaggio PHP. Ebbene, uno dei pochi limiti di quest'ultimo è che il caricamento di nuovi contenuti necessita per forza del refresh della pagina.
Fortunatamente proprio in questo caso, AJAX (acronimo di **Asyncrounous Javascript and XML**) ci viene in aiuto.

Molti commettono l'errore di pensare che AJAX sia un linguaggio a sé stante, mentre in realtà si tratta di diversi linguaggi e componenti uniti tra loro.

AJAX in Javascript

Il linguaggio dove AJAX viene maggiormente utilizzato è senza dubbio Javascript. Esso permette la piena comunicazione tra il client (il browser, per intenderci) ed il server su cui risiede la web-application. Qui sotto puoi vedere un breve snippet di codice relativo ad una chiamata AJAX:

Il codice, specialmente ai neofiti, potrebbe sembrare ostico, motivo per cui cercherò di spiegarlo al meglio.

Innanzitutto, inizializziamo l'oggetto *XMLHttpRequest()* che sarà l'oggetto portante per la comunicazione con il server. Sarà proprio il suddetto a permetterci di inviare la richiesta e controllare la risposta.

Successivamente, al cambiamento di stato dell'oggetto indicato da *onreadystatechange* (solitamente avviene in corrispondenza di un evento del DOM, quando ad esempio vogliamo prendere dei dati da un'altra pagina al click di un input), controlliamo come prima cosa se la risposta è stata correttamente ricevuta, funzione resa possibile grazie all'attributo *readyState*. In seguito, andremo a verificare la risposta del server: il codice **200** indica per l'appunto che tutto è andato a buon fine.

A questo punto ci si potrebbe chiedere perché fare due controlli diversi se già ci assicuriamo che la risposta sia stata ricevuta in precedenza. Come già detto, readyState si assicura che la risposta sia stata ricevuta, ma non controlla che tipo di risposta sia (in quanto potrebbe essere anche una 404,

che indica un **file non trovato**).

Chiusa questa piccola parentesi, andiamo ad aprire la connessione utilizzando il metodo *.open*, che sfrutta in questo caso tre parametri: il primo sta ad indicare il metodo impiegato per i dati, dove *GET* indica per l'appunto l'intenzione di voler estrarre dati dalla pagina utilizzata, la cui URL è inserita come secondo parametro. Il terzo parametro, invece, determina il tipo di richiesta, che può essere **sincrona** o **asincrona**.

L'ultima linea di codice rappresenta l'invio della richiesta al server, cosa che viene resa possibile dal metodo *.send()*.

jQuery e AJAX

Il nostro framework preferito ha messo a punto un sistema molto più veloce (e anche più comprensibile) per inviare una richiesta AJAX, con performance molto simili a Javascript nativo. Vediamo come si utilizza:

Come puoi evincere dal codice appena sopra, jQuery offre una sintassi molto più essenziale. Innanzitutto, dichiariamo il tipo di richiesta, in questo caso di tipo **GET**; successivamente, definiamo l'URL dal quale estrarre i dati. Infine, possiamo usare comodamente due funzioni da impiegare rispettivamente in caso di successo o di errore.

Presta particolare attenzione alla funzione associata a *success*: essa, infatti, usa **data** come parametro e ciò significa che all'interno della funzione potremmo manipolare facilmente i dati estratti per utilizzarli all'interno della nostra pagina.

Un piccolo esempio in AJAX

Quello che vedrai qui di seguito è un piccolo esempio di una web-app davvero basilare realizzata con l'ausilio di AJAX. In particolare, avremo una pagina **index.html** con al suo interno un bottone. Al click di quest'ultimo richiameremo i dati da una seconda pagina **items.html**, che caricheremo poi in un contenitore all'interno della prima pagina.

Se arrivato a questo punto il procedimento ti sembra macchinoso, non preoccuparti, in realtà è più facile di quanto pensi.

Il codice HTML

Il codice di markup è davvero essenziale. Tuttavia, ricordiamoci che nell'header di entrambe le pagine dovremo aggiungere diversi plugin Javascript, iniziando naturalmente da jQuery:

Successivamente, implementiamo gli script GSAP (di cui ti avevo parlato in questo articolo), molto utili per estendere le animazioni del core di jQuery:

Se sei un maniaco delle prestazioni, non hai nulla di cui preoccuparti: il peso complessivo di questi script è di circa **16kb**.

Ora passiamo al contenuto del body, che nel caso della pagina index.html sarà semplicemente:

Mentre nella seconda pagina avrà un aspetto del genere:

Ora iniziamo a dare uno stile più completo alla pagina index.html (la pagina items.html, in questo caso, non è oggetto d'interesse, perché nel nostro esempio non sarà visionabile dall'utente finale).

Il CSS

Nel nostro progetto anche il CSS sarà molto basilare. Tralasciandone il lato puramente decorativo, andremo ad esaminare quello necessario per avere un layout più che discreto.

Come già anticipato, le parti relative al layout sono davvero essenziali. Per quanto riguarda il transform, è stato implementato funzionalmente alle animazioni jQuery che vedremo a breve.

Il codice jQuery

Anche qui, il codice jQuery non sarà nulla di particolarmente complesso e/o astratto, quindi andiamo a vedere lo script completo:

Partiamo subito inizializzando l'evento che riconoscerà il caricamento della pagina, ovvero il *.ready()*. Successivamente, assegniamo la variabile *button* all'effettivo elemento per attuare il caricamento, dopo detectiamo l'evento *.click()* su quest'ultimo, in concomitanza del quale dovrà avvenire la chiamata AJAX. Fatto ciò, iniziamo subito a stabilire le tre proprietà più importanti della chiamata:

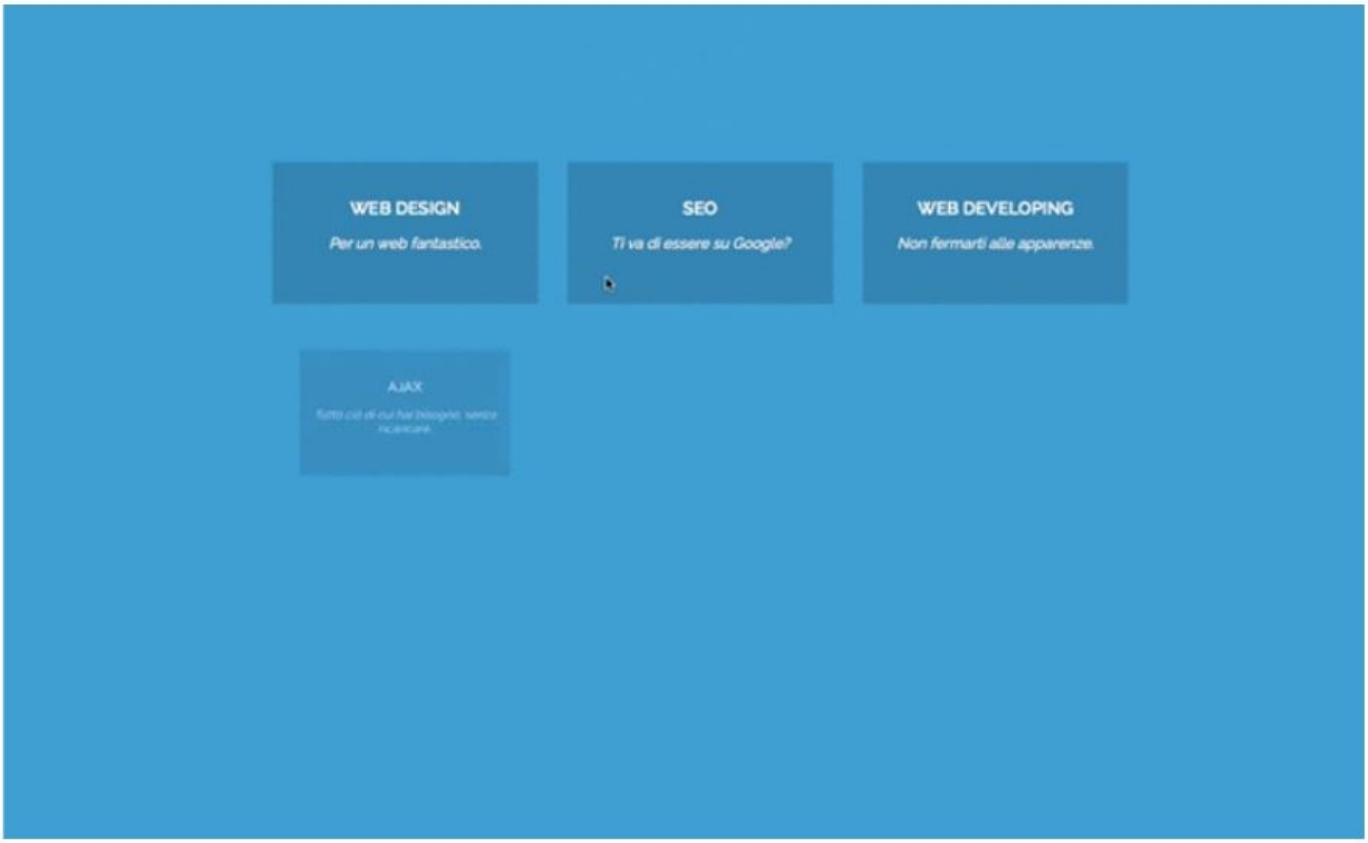
- Il metodo, in questo caso “GET”
- L’url, in questo caso “items.html”
- La funzione **success**, all'interno della quale eseguire le nostre operazioni qualora la pagina items.html sia stata trovata.

A questo punto, possiamo davvero sbizzarrirci a suon di animazioni, giacché spesso e volentieri con questa funzione si manipolano i dati ricevuti, appendendoli al documento corrente (vedasi il parametro 'data', già citato in precedenza).

Fatto ciò, assegniamo la variabile *$els*, che si occuperà di ricavare gli elementi desiderati, aiutata dal metodo *.filter()*.

In seguito, andiamo ad animare il button, nascondendolo con un gradevole effetto di scaling, e appendiamo tutti gli elementi trovati all'interno di *.text-container*, per poi animarli uno ad uno grazie ai metodi *.each()* e *.animate()*.

Nell'immagine qui sotto possiamo vedere lo script in azione.



Conclusioni

Anche l'articolo di oggi è terminato. Ricorda, nei tuoi prossimi progetti, che AJAX è una tecnologia davvero potente. Puoi gestire anche operazioni di aggiunta e rimozione voci (ad esempio nella dashboard di un CMS) senza il bisogno di refreshare continuamente la pagina, e anzi, con

l'aggiunta di animazioni molto ma molto carine.

Come al solito puoi visionare il progetto completo a questo link, e puoi scaricare tutto il codice sorgente impiegato da qui.

Alla prossima ragazzi, e buon lavoro!