Introduzione

Le moderne applicazioni web, intendiamo quelle che hanno potuto beneficiare di un budget adeguato per lo sviluppo, possono raggiungere livelli di qualità significativamente elevati. Addirittura, se le stiamo utilizzando su un dispositivo dotato di una buona connessione di rete e di sufficienti risorse hardware, possono offrire un'esperienza d'uso paragonabile a un'applicazione nativa.

Esempi di applicazioni web con questo livello di finitura (solo per citarne alcuni) sono Gmail di Google (`https://mail.google.com`), iCloud di Apple (`https://www.icloud.com/`) e Wunderlist di Wunderkinder (`http://www.wunderlist.com/home`).

Le applicazioni che raggiungono questi picchi di qualità riescono grazie a un sapiente uso di HTML, CSS e JavaScript (oltre ad Ajax e Photoshop). Ma il livello tecnico richiesto è molto alto, perché per realizzare applicazioni di questo livello è necessario affrontare problematiche complesse (oltre che multidisciplinari).

Rimanendo nel contesto della pura programmazione software e limitando il discorso alle tecnologie di base, la prima sfida che lo sviluppatore si trova ad affrontare è l'eterogeneità dei software utilizzati dagli utenti per fruire dell'applicazione: i browser.

Un florilegio di browser

Un browser molto utilizzato oggigiorno è Firefox di Mozilla Foundation, che si contende quote di mercato con quello attualmente più diffuso: Internet Explorer di Microsoft. Ma a fianco di questi due concorrenti ci sono altre alternative, anche se con una diffusione minore: Safari di Apple, Chrome di Google e Opera di Opera Software. Ciascuno di questi browser ha proprie caratteristiche peculiari, che in termini pratici si traducono in incompatibilità tra di essi e con gli standard internazionali dettati dal W3C (quelli che tutti invece dovrebbero rispettare alla lettera).

In particolare, Microsoft è sempre stata accusata di scarsa attenzione al rispetto degli standard, e lo sviluppo con Internet Explorer 5 e 6 si è rivelato in passato decisamente problematico. Con l'uscita delle versioni 7, 8 e 9 le cose sono migliorate, ma ancora oggi siamo lontani da una piena compatibilità e anzi, specifiche CSS ormai molto datate continuano a non essere supportate correttamente nei browser di casa Microsoft. Dal lato degli outsider rileviamo che Safari e Chrome condividono lo stesso motore di rendering, WebKit (`http://www.webkit.org/`, un progetto open source), elemento che agevola senza dubbio la compatibilità tra i software e il lavoro degli sviluppatori.

Firefox e Opera hanno invece motori proprietari, il secondo spinto anche dalla necessità di offrire funzionalità peculiari allo scopo di differenziarsi dalla concorrenza e giustificare la propria esistenza (Opera è un software a pagamento, mentre tutti gli altri browser citati sono aperti o gratuiti).

Questo è il panorama che lo sviluppatore si trova ad affrontare, ma in realtà la situazione è più complessa. Si consideri infatti che l'insieme di questi browser è disponibile sul sistema operativo Windows, ma che la maggior parte di essi ha declinazioni anche per Mac OS X e Linux, i due principali sistemi utilizzati per i computer desktop. Anche se la compatibilità tra le versioni su piattaforme diverse è decisamente migliore che in passato, rimane che si tratta comunque di software differenti e che quindi richiederebbero test appositi. Inoltre va ricordato che sistemi operativi specifici offrono browser specifici, come Camino e OmniWeb per Mac OS X, o Konqueror o Epiphany per Linux.

Facendo un conto approssimativo (cinque browser per tre piattaforme), siamo già a 15 combinazioni che un solerte gruppo di sviluppo dovrebbe testare singolarmente prima di rilasciare la propria applicazione. Bisogna poi considerare l'esistenza – e quindi conteggiare – le diverse tipologie e versioni di sistema operativo (Windows XP, Windows Vista, Windows 7 e relativi Service Pack, Mac OS X Snow Leopard, Mac OS X Lion e tutte le varianti di Linux). In concreto, questo non succede mai, perché il costo sarebbe spropositato; nel mondo reale si sceglie un ventaglio di possibilità molto più ristretto e si concentrano i test dell'applicazione sulle combinazioni browser/sistema operativo selezionate.

Web in mobilità

In realtà la situazione è ancora più articolata, dato che esistono anche i browser integrati nei dispositivi mobili, come smartphone e tablet.

Dall'avvento di iPhone nel 2008 (anno di uscita in Italia, negli Stati Uniti è uscito l'anno prima), l'accesso a Internet in mobilità ha avuto un'importante accelerazione. La nuova classe di dispositivi che è nata come conseguenza delle innovazioni di Apple ha affermato come normalità la navigazione tramite dispositivi mobili in forma confrontabile a quanto disponibile sul desktop (in contrapposizione alla sua declinazione impoverita e limitata chiamata WAP). Sempre più persone utilizzano dunque dispositivi mobili per accedere ad applicazioni web e ciascun dispositivo spesso ha una propria specifica variante di browser.

Apple produce tre dispositivi che possono accedere alla Rete: iPhone, iPod touch e iPad. Le risoluzioni schermo sono due (più due varianti Retina), e attualmente sono utilizzate perlopiù due versioni di iOS, la 4 e la 5. Si tratta in definitiva di otto varianti.

Nel mondo Android il numero di dispositivi è decisamente più elevato, come la quantità di versioni software. Da alcune stime (`http://tnw.co/H1lWU2`) si parla di più di 1400 combinazioni possibili tra tipo di dispositivo (quindi risoluzione dello schermo) e versione del sistema operativo (ovviamente non tutte influiscono in modo significativo sulla navigazione web). Alcune rilevazioni arrivano fino a superare la quota 3000 (`https://twitter.com/#!/blackjack75/status/193764950290862080`).

Nel resto del comparto mobile è bene segnalare la ripresa di Windows Mobile con la versione 8 (ma la quota di mercato è ancora piccola, come del resto il numero di dispositivi che lo supportano) e la quasi totale scomparsa di BlackBerry.

Bisogna infine considerare che la navigazione con iPad (o con altri tablet) è molto simile a quella che avviene con un desktop con scheda grafica un po' obsoleta (la risoluzione del tablet di Cupertino è 1024×768, anche con i meravigliosi nuovi display Retina, che si limitano a presentare le immagini al doppio della risoluzione, mantenendo l'area di visualizzazione del sito a 1024×768), ma non è impossibile assimilare questo device a un computer desktop, se si esclude la tecnologia Flash, che sui dispositivi di Apple non è supportata e che Adobe sta abbandonando anche sulle altre piattaforme (`http://blogs.adobe.com/conversations/2011/11/flash-focus.html`).

La navigazione con smartphone, diversamente dai tablet, spesso prende una strada a sé, con siti web costruiti appositamente per la dimensione contenuta del display, siti che talvolta non sono in grado di offrire l'insieme delle funzionalità presenti nella versione per desktop, ma

che spesso sono costretti, per questioni di budget, a limitarsi alle sole funzionalità principali.

Applicazioni multibrowser

In questo panorama articolato e frammentato, dire che un'applicazione è *multibrowser* è sicuramente un azzardo, fosse anche solo perché è materialmente impossibile eseguire dei test su tutte le varianti di dispositivi/browser presenti sul mercato. In realtà, i progetti web che vengono realizzati oggigiorno sono certificati per un ristretto insieme di combinazioni di sistema operativo/versione, browser e versione/dimensione schermo.

Ma anche all'interno di un panorama semplificato da una strategia oculata, è necessario gestire correttamente le differenze tecniche tra i diversi browser, attività che può risultare comunque alquanto complessa.

JavaScript

Quando si parla di applicazioni web, si fa riferimento a software che sono costituiti, oltre che da un'importante parte di presentazione e grafica, (codice HTML, CSS e immagini), da una consistente componente di logica applicativa, che nel migliore dei casi è suddivisa in due parti: quella relativa alla presentazione gira sul browser, mentre quella di back-end è ospitata dal server applicativo. Se in quest'ultimo si trova codice Java, .NET o PHP, sul browser il linguaggio di riferimento è JavaScript.

Quest'ultimo è un linguaggio di scripting di cui si può abusare facilmente, arrivando a scrivere codice complesso e ingestibile. Negli anni più recenti, però, si sono affermati stili di programmazione più moderni, che consentono anche di scrivere codice orientato agli oggetti (JavaScript supporta l'*object orientation* tramite prototipazione, la programmazione funzionale e quella imperativa: `http://en.wikipedia.org/wiki/Javascript#cite_note-jsfunc-0`).

Inoltre, la qualità degli interpreti JavaScript più recenti è molto migliorata in termini di performance e affidabilità (anche nel comparto *mobile*), fatto che rende più funzionale il codice eseguito sul browser; la maggior performance, anche grazie a processori più potenti, può essere spesa per funzionalità non essenziali, come effetti grafici e anima-

zioni, che prima dovevano essere affidati a tecnologie differenti, come immagini animate o Flash.

Questi sviluppi, se da un lato rendono possibile migliorare l'esperienza d'uso applicativa tramite feedback animati e un'interazione più evoluta (come il controllo in linea della validità dei campi o validazioni server tramite Ajax), rendono il lavoro di sviluppo più complesso. Da una parte c'è la necessità di ottenere la medesima funzionalità su tutti i browser, che per via delle incompatibilità tra gli stessi richiede accorgimenti particolari; dall'altra diviene imperativo, quando si realizzano applicazioni di entità non banale, la realizzazione di codice pulito e ben architettato, al fine di evitare incubi in fase di manutenzione dello stesso.

Questo fa a pugni con alcune caratteristiche di JavaScript, che non sono abbastanza amichevoli, e con il modo in cui si interagisce dal codice con la struttura del documento caricato nel browser, la pagina. L'accesso a quest'ultimo avviene infatti attraverso un insieme di API il cui uso non è affatto banale. Queste API sono l'interfaccia pubblica del *Document Object Model* (DOM), un insieme di convenzioni multipiattaforma e indipendenti dal linguaggio che rappresentano oggetti in documenti HTML, XHTML e XML e che ne consentono la manipolazione. La natura fortemente strutturata e gerarchica di questa tipologia di documenti si riflette sulla struttura del DOM, la cui interrogazione e modifica richiede verbose operazioni di iterazione e ricorsione. Il necessario traversamento dell'albero che rappresenta gli oggetti del documento può quindi richiedere una quantità significativa di codice, che pregiudica non tanto e soltanto le performance in fase d'esecuzione del codice, ma soprattutto la leggibilità e la pulizia del codice.

jQuery

Per ovviare a questi inconvenienti è possibile utilizzare una delle diverse librerie JavaScript disponibili in Rete, che racchiudono funzionalità studiate per superare queste limitazioni.

Una delle più affermate, e la protagonista di questo libro, è jQuery (`http://jquery.com/`). L'home page del progetto afferma che si tratta di "una libreria JavaScript veloce e concisa che semplifica il traversamento dei documenti HTML, la gestione degli eventi, l'animazione e le interazioni Ajax per uno sviluppo web rapido".

Al di fuori delle parole dei suoi stessi autori, jQuery è comunemente considerato uno dei protagonisti della rinascita della tecnologia Java-

Script, soprattutto per via di tre caratteristiche fondamentali: la buona capacità di nascondere le peculiarità dei browser offrendo un'interfaccia e funzionalità omogenee; la possibilità di accedere a elementi della struttura DOM attraverso selettori CSS e la semplicità d'uso.

jQuery è una libreria JavaScript e su questo si concentra; in più, le sue componenti *core* sono decisamente a basso livello. Non siamo di fronte a un framework di funzionalità ad alto livello che consentono di sviluppare applicazioni in modo completamente indipendente dal browser: problematiche legate alle differenze tra desktop e mobile o alla presentazione grafica (HTML e CSS) sono al di fuori del suo campo d'azione.

jQuery è una libreria compatta per design; sebbene in poco tempo sia cresciuta di quasi il doppio (nel 2010 il peso della libreria era di appena 19 KB), il team che ne gestisce lo sviluppo è accorto a selezionare con attenzione quello che va inserito nella libreria di base e quello che deve essere invece oggetto di un plug-in esterno.

Tecnologie moderne

Con jQuery è possibile utilizzare tecnologie all'ultimo grido, anche con browser antiquati che non le supportano direttamente. È il caso di CSS3: anche se il browser non implementa questo livello di specifica, jQuery è in grado di capirlo comunque, grazie a una propria implementazione interna. Ovviamente il semplice fatto di includere jQuery in una pagina web non significa che il browser acquisisce la capacità di capire fogli di stile esterni con sintassi CSS3. Ma se rimaniamo nel contesto dei soli selettori utilizzati specificatamente con funzioni jQuery è possibile sfruttare anche CSS3.

Inoltre, jQuery offre la possibilità di elencare le funzionalità avanzate (HTML5 e così via) supportate dal browser, in modo che il proprio codice possa agire di conseguenza; dove possibile, cerca di ovviare alle mancanze del browser implementando internamente le funzionalità mancanti.

Ampia adozione

Quando si sta selezionando un componente software o una libreria da integrare nel proprio progetto, uno degli elementi che può far propendere verso una decisione positiva o negativa è il grado d'affermazione

del candidato. Si tratta qualcosa di appena uscito? Forse non è stato testato abbastanza. Il rischio di affidarsi a una tecnologia non provata può essere un elemento critico in un progetto software. Se il software è disponibile da tempo, quante aziende lo usano e quali? Il numero e la qualità dei *referral* è utile per determinare il grado di solidità e stabilità di una data soluzione.

jQuery è una libreria affermata sulla Rete da diversi anni e vanta il fatto di essere utilizzato da molte aziende e organizzazioni di primaria importanza, come Google, Dell, Bank of America, NBC, CBS, Netflix, Technorati, Mozilla.org, WordPress e Drupal.

Questi sono ovviamente solo i riferimenti più importanti, presenti sull'home page del sito ufficiale della libreria.

Ordine e semplicità

Utilizzare jQuery nel proprio sito o applicazione web significa promuovere una organizzazione più strutturata del progetto.

La possibilità di accedere facilmente agli elementi della pagina HTML attraverso selettori CSS rende possibile raccogliere le funzionalità d'interazione in un punto solo (lo script) invece che distribuirli per tutta la pagina. In altre parole, con jQuery non è necessario disseminare il documento HTML con attributi `onClick` o `onSubmit`, singolarmente e per ciascun elemento che lo richiede.

È invece possibile, in un blocco `<script>` specifico, oppure e ancor meglio, in un file JavaScript separato, identificare gli elementi da modificare e, anche in gruppo, assegnare i gestori d'evento necessari.

Le declinazioni di jQuery

Oltre alla libreria jQuery di base, che contiene il cuore delle sue funzionalità, nell'universo jQuery sono presenti altre tre componenti.

- I plug-in (`http://plugins.jquery.com/`) sono componenti aggiuntivi costruiti sulle funzionalità di base di jQuery. Questi implementano effetti aggiuntivi, widget, componenti Ajax, tabelle e molto altro. Al momento in cui scriviamo il sito non è disponibile, in quanto in fase di rifacimento. Gli archivi sono comunque presenti all'indirizzo `http://archive.plugins.jquery.com/`.
- jQuery User Interface (`http://jqueryui.com/`) è invece una libreria open source di componenti per l'interfaccia utente. Il progetto

fornisce astrazioni a basso livello per l'interazione e l'animazione, come effetti avanzati e widget con supporto ai temi per la realizzazione di interfacce ad alto livello. Tra i widget troviamo il menu a fisarmonica (*accordion*), il campo con autocompletamento, il pulsante, il selettore di date, la finestra di dialogo, la barra di progressione, lo slider e il contenitore a schede. È possibile scaricare jQuery UI (`http://jqueryui.com/download`) in forma completa oppure selezionando le sole componenti di cui si necessita, per ottimizzare la dimensione del file.

- jQuery Mobile (`http://jquerymobile.com/`) è la declinazione di jQuery e jQuery UI per dispositivi mobili. Si tratta di un framework web ottimizzato per i dispositivi sensibili al tocco come smartphone e tablet. È un sistema di interfaccia utente basato su HTML5 e unificato per funzionare su tutte le popolari piattaforme per dispositivi mobili (e anche le meno popolari), come iOS, Android, BlackBerry, Windows Phone, Palm webOS e Symbian. Dato che non tutti i dispositivi sono uguali, jQuery Mobile non utilizza l'approccio del *minimo comune denominatore*, ma un sistema a profili: quello più alto include la *user experience* completa con animazioni basate su Ajax, quella intermedia esclude Ajax, mentre quella più bassa ha le funzionalità minime ma sufficienti perché l'applicazione sia operativa. Come è facile immaginare, i dispositivi più recenti offrono un'esperienza d'uso completa, mentre quelli più datati vengono gestiti attraverso i profili funzionali più limitati. Come la controparte desktop, è una libreria leggera, ma pensata per poter essere estesa. Implementa inoltre un sistema di temi grafici flessibile e personalizzabile.

Scopo del libro

Lo scopo di questo testo è quello di proporre al lettore una guida approfondita delle funzionalità di base di jQuery, sfruttando numerosi esempi per illustrare le caratteristiche salienti di questa libreria.

Anche se il libro non ha lo scopo di essere completo al 100%, si è cercato di coprire il più possibile i gruppi di funzionalità più importanti, in modo che il testo possa essere utilizzato anche come *reference*.

Sono fuori dagli scopi di questo libro le funzionalità più avanzate e interne di jQuery, come quelle necessarie per realizzare plug-in o altre estensioni della libreria.

Prerequisiti

Per poter usufruire al meglio del testo è necessario avere una conoscenza anche basilare del funzionamento del World Wide Web e del protocollo HTTP, come anche di HTML, CSS DOM e JavaScript. Anche se non è indispensabile, può far comodo conoscere un po' di Ajax.

Contenuti

Il libro si articola in otto capitoli.

- Capitolo 1, "Cos'è jQuery". Introduce alle caratteristiche della libreria e mostra come scaricarla e collegarla alle pagine HTML. Illustra come utilizzare i Content Delivery Network e fornisce i primi esempi d'utilizzo della librerie.
- Capitolo 2, "Selettori". Mostra come utilizzare i selettori di jQuery per raggiungere facilmente qualsiasi elemento della pagina HTML, utilizzando anche espressioni complesse e composte, che prendono in considerazione il tipo, la classe, l'ID e gli attributi dell'elemento.
- Capitolo 3, "Filtri". Il risultato di una selezione può essere ulteriormente ristretto attraverso un corposo insieme di filtri; questo capitolo li elenca e li descrive.
- Capitolo 4, "CSS". Elenca e descrive tutte le funzioni offerte da jQuery per operare con classi e proprietà CSS.
- Capitolo 5, "Animazione". Illustra i metodi predefiniti per l'animazione di elementi HTML, come lo scorrimento e le dissolvenze. Introduce poi le animazioni personalizzate, alcuni parametri di configurazione e le code di animazioni, che consentono di creare effetti complessi aggregando transizioni elementari.
- Capitolo 6, "Eventi". Descrive i metodi di gestione degli eventi di jQuery, che costituiscono uno strumento semplice e potente per poter collegare funzioni di gestione agli eventi JavaScript. Inoltre, descrive le caratteristiche implementate dalla libreria per uniformare le differenze tra i vari browser nella gestione degli eventi.
- Capitolo 7, "Traversare e manipolare il DOM". Affronta tutti i metodi messi a disposizione da jQuery per navigare nella struttura gerarchica degli oggetti che rappresentano la pagina e i relativi attributi. Descrive inoltre i metodi che è possibile utilizzare per cambiare il contenuto degli elementi.

- Capitolo 8, "Ajax". Affronta le funzionalità offerte dalla libreria per implementare chiamate asincrone Ajax.

La relazione con altri Pocket

Come si vedrà nel paragrafo successivo, l'autore di questo libro ne ha scritti diversi altri. Leggendoli, si troverà sempre qualcosa in comune, talvolta solo per il fatto che la penna è la medesima, altre volte per via della contiguità degli argomenti. Ne è un esempio il trittico sulle tecnologie .NET: *C# 4.0* e *Visual Basic 10* sono due libri speculari, che trattano gli stessi argomenti declinati in due linguaggi di programmazione differenti. *ASP.NET 4.0* si integra con questi due volumi, condividendo alcuni contenuti di base.

Non è la prima volta che l'autore scrive di jQuery in un testo della collana Pocket edita da Apogeo; se si guardano le appendici di *Ajax* si troverà una paginetta dedicata all'argomento del libro che state tenendo in mano. In un certo senso, *jQuery* è la prosecuzione di *Ajax*. Quest'ultimo parte dalle basi del World Wide Web e introduce XML, HTML, CSS, DOM e JavaScript, prima di approfondire diverse tecniche Ajax. Il libro che state leggendo dà tutte queste cose per scontate (se serve un'introduzione, *Ajax* è un possibile candidato) e inizia direttamente con la descrizione della tecnologia jQuery.

Convenzioni utilizzate nel libro

Nel libro sono presenti alcune icone che identificano tipi particolari di informazioni relative agli argomenti trattati.

Terminologia *Questa icona segnala o spiega termini nuovi o poco familiari.*

Nota *Una nota contiene informazioni interessanti, talvolta tecniche, relative all'argomento trattato. Può riportare approfondimenti e curiosità.*

Attenzione *I paragrafi contrassegnati da questa icona presentano annotazioni di particolare rilevanza.*

Suggerimento *Questa icona contrassegna spunti e consigli per risparmiare tempo ed evitare confusioni, come per esempio il modo più semplice per eseguire una determinata operazione.*

Codice degli esempi

All'indirizzo `http://www.apogeonline.com/libri/9788850331444/scheda` è possibile scaricare un archivio in formato ZIP contenente i listati di codice utilizzato negli esempi, organizzato in capitoli.

L'autore

Enrico Amedeo inizia a programmare negli anni Ottanta su computer Commodore. Nel 1990 realizza il suo primo programma commerciale su processore Z80. Scrive di tecnologia dal 1999, e ha pubblicato centinaia di articoli e numerosi libri. Per Apogeo ha firmato nella collana Pocket i volumi *Ajax*, *C*, *UML*, *C# 4.0*, *Visual Basic 10*, *ASP.NET 4.0* e *Objective-C*. È autore, insieme a Federico Gatti, di *iPad* e *iPad2*, editi entrambi nella collana Pocket Color.

Nel tempo libero gli piace fotografare e praticare Kung fu.