Nuove prospettive per un uso didattico di Internet

" There's never been a greater time for the software industry…"
Bill Gates, dicembre 1995

Luigi Sarti
Ricercatore
ITD-CNR, Genova
sarti@itd.ge.cnr.it

Stiamo assistendo ad una rapida evoluzione delle tecniche di presentazione del dato ipermediale in Internet. Dal punto di vista dell'utente le pagine si fanno sempre più interattive, veloci nel caricamento, ricche di funzionalità e simili agli ipermedia distribuiti su disco o CD-ROM e navigabili *in loco*, senza coinvolgere il canale telematico. Dal punto di vista dell'autore di pagine Web, per altro, le architetture e gli strumenti di sviluppo sono sempre più sofisticati: è importante poter aggirare le limitazioni intrinseche del linguaggio HTML e soddisfare i requisiti di scalabilità, rapidità di sviluppo e integrabilità di dati esterni che con sempre maggiore frequenza emergono. In questo scenario cambia radicalmente anche la prospettiva di chi si propone di usare Internet a scopi didattici: la struttura ipertestuale del Web si presta particolarmente alla distribuzione di materiale ipermediale, ed ulteriore benefici derivano dalla standardizzazione dei meccanismi di presentazione, che agevola la fruizione su piattaforme e sistemi diversi e fino a ieri incompatibili. Le recenti innovazioni tecnologiche offrono inoltre un ampio spettro di funzionalità e strumenti all'autore di *courseware*, e contribuiscono al soddisfacimento di una gamma di esigenze già evidenziate dalle prime esperienze in questo settore.

LE ESIGENZE

Secondo una rilevazione dell'Osservatorio Alchera[1], al Settembre 1997 gli utenti Internet nel nostro Paese sono diventati oltre 2.300.000, con una crescita di un milione di unità negli ultimi sei mesi [Mandò 97, Mandò 98]. I risultati di una recente ricerca riportata in [Livraghi 98] sono invece meno eclatanti: saremmo oggi intorno agli 800 mila utenti o poco più, di cui meno di 400 mila si collegano da casa e più di 700 mila hanno un collegamento sul luogo di lavoro. I dati precedenti sono fortemente discordanti, perché tutto dipende dalla definizione di chi è l'utente. Rimane il fatto, condiviso da tutti i ricercatori, che l'utilizzo di Internet in Italia è in forte crescita. Negli Stati Uniti la diffusione di Internet sembra avvicinarsi alla saturazione: mentre nel '96 la crescita aveva raggiunto il 70% annuo, nel '97 si assesta sul 30%; quasi 37 milioni di PC risulta con regolarità collegati ad Internet nell'Agosto '97; circa il 50% dei PC acquistati nel '97 viene collegato [Broersma 97].

L'improvvisa ed enorme diffusione dell'uso di Internet ha causato negli ultimi tempi una crescita delle aspettative dell'utenza, che richiede funzionalità sempre più sofisticate. Secondo il rapporto annuale della *Price Waterhouse*, Internet ha assunto un ruolo chiave nel determinare le linee di evoluzione delle tecnologie informatiche [Brethour 97]. Netscape ha rilasciato tre versioni del suo browser in 18 mesi. Microsoft ha fatto lo stesso in 15 mesi. Chi sviluppa software fruibile in Internet è costretto a tenere ritmi simili aggiornando i propri prodotti in media ogni sei mesi, pena l'esclusione dal mercato.

Fino a pochi mesi fa la maggior parte dei siti WWW si limitavano ad offrire un insieme di pagine statiche, navigabili in modo ipermediale secondo percorsi precostituiti anch'essi staticamente; la modalità d'interazione consentita al di là della semplice navigazione si basava prevalentemente sulla compilazione, da parte dell'utente, di formulari che consentono il trasferimento di dati dal cliente al *server* (vedi Figura 1). Se la meccanica dell'interazione richiede uno scambio bidire-

zionale e costante di messaggi tra cliente e *server*, il ritardo tra ogni azione dell'utente e la conseguente risposta del sistema è spesso inaccettabile, soprattutto quando il server debba far fronte a numerose connessioni contemporanee, come nel caso di siti frequentemente visitati. Chi ha provato a connettersi con un sito molto "trafficato" ha sicuramente sperimentato questa situazione: ai ritardi intrinseci della trasmissione telematica, spesso realizzata alle basse velocità delle connessioni via modem, si sommano quelli dovuti al sovraccarico del *server*. Si consideri inoltre che nello scenario sopra descritto la presentazione di dati multimediali intrinsecamente dinamici come animazioni, segmenti sonori e sequenze filmate si basa sempre sul trasferimento di files dal server al cliente, e sulla successiva fruizione di questi mediante specifiche applicazioni opportunamente configurate nel *browser*: una soluzione di questo tipo impone tempi d'attesa spesso intollerabilmente lunghi, perché l'intero oggetto multimediale deve essere trasferito in toto sulla postazione locale attraverso connessioni spesso lente, prima che la sua fruizione possa aver inizio.

In quella prima fase dell'utilizzo massiccio di WWW si sono manifestate quindi le seguenti esigenze, per lo più percepite come carenze dei sistemi in uso:

- rapidità di risposta
- reattività
- scalabilità
- rapidità di sviluppo

Nel seguito queste esigenze verranno esaminate in maggior dettaglio.

Rapidità di risposta

L'utente vuole poter interagire con la pagina WWW come se questa fosse un'applicazione presente localmente sul suo PC. Premo un bottone, e voglio subito una risposta: sono disposto ad aspettare solo a fronte di elaborazioni che riconosco come particolarmente onerose dal punto di vista computazionale, ma se devo aspettare diversi minuti per poter vedere un filmato che dura poi meno di un minuto, preferisco rinunciare.

Reattività

Spesso le funzionalità di presentazione ipermediale tradizionalmente offerte dal Web risultano insufficienti: non basta visualizzare materiale multimediale navigabile con varie modalità, l'interazione con l'utente deve essere supportata da funzionalità computazionali di vario tipo: i dati immessi nei formulari devono essere accettati, validati e registrati nel sistema server; talvolta le pagine non sono registrate staticamente nel *file system* del server, ma devono essere costruite dinamicamente, con dati provenienti da fonti

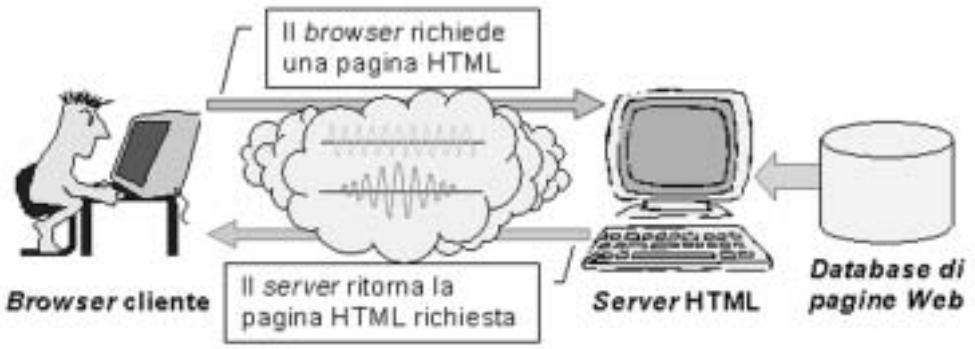


Figura 1
Il modello client-server.

Le pagine Web sono interconnesse da una fitta rete di riferimenti ipertestuali (le *ancore*). Ogni ancora contiene in sé il nome della pagina riferita, e quello del *server* che la amministra. Ogni volta che da un *browser* viene attivato un collegamento ipertestuale, un messaggio di richiesta parte alla volta del *server* responsabile; questi recupera dal proprio archivio la pagina desiderata e la inoltra al richiedente unitamente alle eventuali componenti multimediali che ne fanno parte. Lo scambio di messaggi avviene secondo le regole di uno specifico protocollo, denominato *http*. Nell'architettura Internet tradizionale tutti i comportamenti interattivi delle pagine Web vengono ricondotti ad una transizione telematica di questo tipo.

svariate (es. database). In campo didattico, poi, la necessità di pagine Web che sappiano anche elaborare dati è particolarmente sentita, ad esempio quando si vogliano integrare funzionalità di simulazione o test diagnostici.

Scalabilità

Col termine *scalabilità* si intende la capacità di usare lo stesso ambiente software su una varietà di configurazioni e piattaforme. Se, a fronte di un forte incremento della base d'utenza, le prestazioni di un sistema possono essere mantenute costanti semplicemente rinforzandone la piattaforma hardware, quel sistema esibisce buone doti di scalabilità. Oltre un certo livello, tuttavia, arricchire le risorse hardware può non bastare più: in tal caso si rende necessario il ridisegno dell'architettura software.

Si calcola che alla fine del 1996 Netscape venisse usato in tutto il mondo da quasi 50 milioni di utenti [Barksdale 96]. Nello stesso periodo i siti più gettonati negli USA vantano frequenze d'accesso da capogiro: *MapQuest*[2] è usato da 2 milioni di utenti al giorno; il sito della *SouthWest Airline*[3] consente a 250.000 utenti al giorno di prenotare voli e acquistare biglietti; il sistema *Pathfinder*[4] di *Time-Warner* genera dinamicamente quasi mezzo milione di pagine all'ora. Secondo *Andromedia* i livelli di attività esibiti all'inizio del 1997 dai primi dieci siti al mondo diventeranno valori medi entro la metà del 1998 [Palmer 97]. Di fronte a questa prospettiva, è comprensibile che i progettisti dei siti si pongano il problema di come garantire tempi medi di servizio accettabili.

Dal punto di vista didattico questi aspetti possono sembrare secondari: in genere chi appronta e mantiene siti Web ad uso formativo non si confronta con carichi computazionali di queste dimensioni; tuttavia la problematica può presentarsi, su scala minore, anche con soltanto qualche decina di utenti, se questi tendono a concentrare gli accessi ad uno stesso sito in un breve lasso di tempo (come potrebbe fare una classe scolastica durante l'ora di laboratorio).

Rapidità di sviluppo

I livelli di competitività raggiunti oggi dal mercato del software per il personal computing costringono i produttori ad accorciare sempre più i tempi di sviluppo delle applicazioni. Per evitare che la qualità del prodotto ne risulti degradata cresce la domanda di metodiche e strumenti di sviluppo sempre più sofisticati che, privilegiando la facilità d'uso, offrano al progettista meccanismi visuali di interazione e promuovano il riutilizzo di componenti precostituiti.

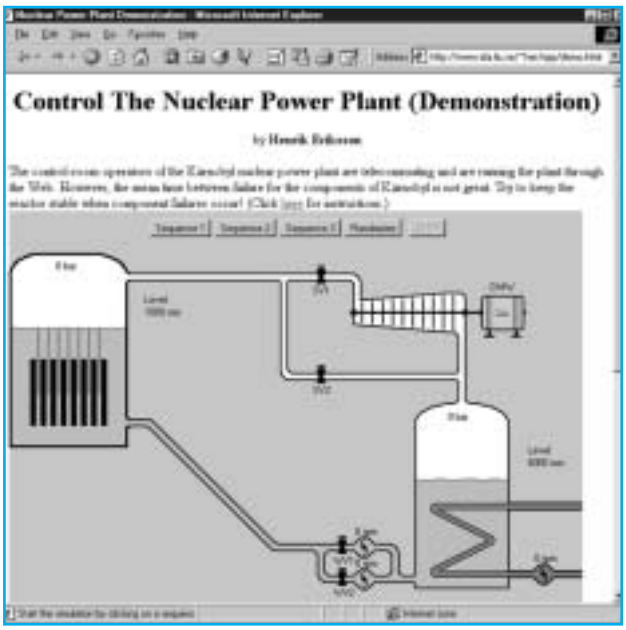
LE NUOVE TECNOLOGIE PER INTERNET

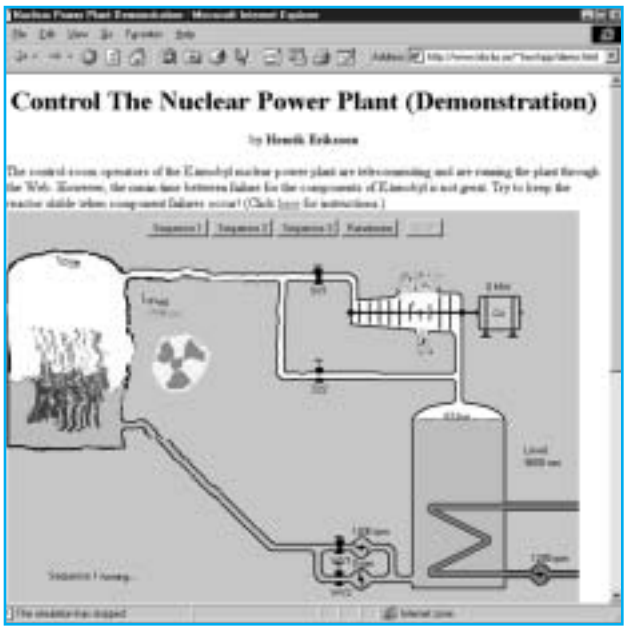
Java

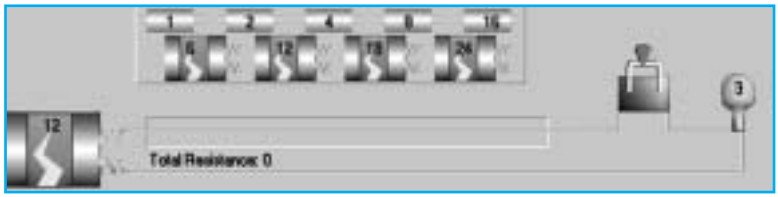
Disegnato dalla Sun Microsystems, Java è un linguaggio di programmazione orientato agli oggetti e particolarmente adatto al *network programming*. Sebbene sia possibile in Java sviluppare applicazioni tradizionali, il suo uso più frequente è rivolto allo svilup-

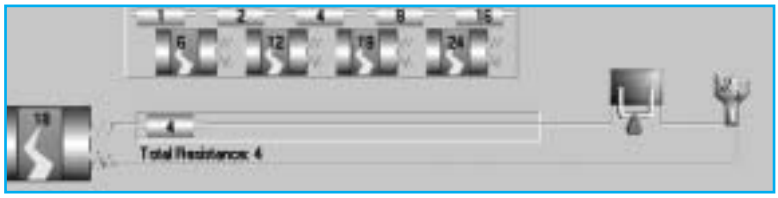
Figura 2
Simulazione di una centrale nucleare.

Figura 3
Il guasto provocato dalla rottura di una pompa.









po di *applet*, mini–applicazioni che possono essere dinamicamente trasferite insieme con una pagina Web ed eseguite dal *browser* cliente. Per consentire ciò i progettisti di Java hanno posto particolare attenzione all'indipendenza dall'architettura, alla sicurezza ed alla robustezza del linguaggio. Grazie agli *applet* le pagine Web possono oggi presentare animazioni, segmenti sonori e in generale comportamenti interattivi in tempo reale.

Secondo una recente indagine [Sun Microsystems 97], a poco più di due anni dalla sua nascita Java può già contare su una base di oltre 450.000 programmatori professionisti nel mondo; più di un milione di applicazioni Java sono state rilasciate, sono stati pubblicati più di 800 libri sul tema e oltre 200 università offrono corsi su Java.

Dal punto di vista didattico Java offre notevoli opportunità: sono infatti disponibili in rete numerosi *applet* che realizzano simulazioni, esercizi, test specificamente disegnati per un uso didattico. Come primo esempio riportiamo la simulazione di un guasto in una centrale nucleare[5], ideata da Henrik Eriksson della Linköping University in Svezia (Figura 2).

L'utente può scegliere fra tre differenti scenari di guasto, e deve tentare di salvare l'impianto operando sulle pompe, sulle valvole e sui livelli delle barre di controllo (Figura 3). Questo *applet* costituisce un eccellente esempio di uso di grafica, animazione, audio e comportamento interattivo.

Altri interessanti esempi di *applet* usabili in contesto educativo sono disponibili nel *Laboratorio Virtuale* dell'Università dell'Oregon[6], dove vengono utilizzati come materiale di supporto nei corsi di Fisica: la Figura 4

Figura 4
La simulazione di un circuito elettrico.

Figura 5
Il circuito nello stato iniziale.

Figura 6
Se la resistenza è insufficiente, la lampadina si brucia.

Figura 7
Con una resistenza adeguata la lampadina si accende.

Figura 8
La pagina introduttiva di Perfect Match.

Figura 9
Esempi di esercizi di Botanica sviluppati con Perfect Match.

Figura 10
Un esercizio di riconoscimento foglia-albero.

Figura 11
Lo studente ha associato correttamente due foglie ai nomi dei relativi alberi.

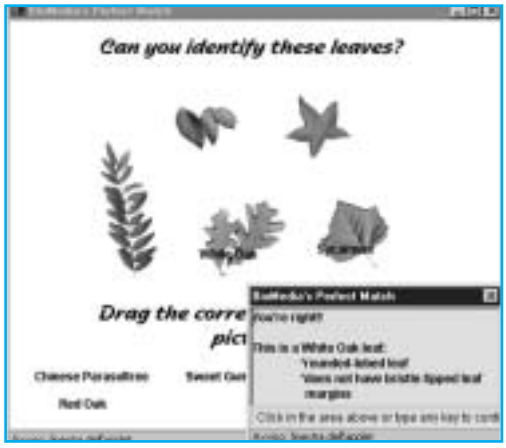


Figura 12
Il test di inglese on line.

mostra la pagina introduttiva di un'unità didattica dedicata alla Legge di Ohm[7], che propone allo studente un semplice esercizio: inserire nel circuito di Figura 5 uno o più resistori (in serie) tali per cui l'intensità di corrente che fluisce nella lampada assuma un valore prestabilito. In dipendenza del valore dell'intensità di corrente la lampada rimane spenta, si brucia (Figura 6) o si accende correttamente (Figura 7). Lo studente può liberamente inserire e togliere resistori, modificare la tensione delle batterie e manovrare l'interruttore di accensione della lampada.

Perfect Match[8] è un altro sito che raccoglie interessanti *applet* utilizzabili in campo didattico. Curato dal *BioMedia Center for Instructional Computing* della *Purdue University*, il sito offre un'ampia raccolta di esercizi in materie scientifiche e mette a disposizione gratuitamente uno strumento per lo sviluppo di nuovi esercizi. In Figura 8 è (parzialmente) visibile la pagina introduttiva di *Perfect Match*; la pagina in Figura 9 elenca gli esempi disponibili in materia di Botanica: selezionando l'esercizio "*Leaf Exercise # 1*" allo studente vengono presentate le forme di alcune foglie, e un elenco di nomi di alberi da mettere in relativa corrispondenza (Figura 10). Quando trascina con il mouse il nome dell'albero sul disegno della foglia corretta, lo studente ottiene un feedback positivo (Figura 11).

Per concludere questa breve serie di esempi, vogliamo citare un *applet* sviluppato in Italia a cura del *British Schools Group*[9], che mette a disposizione un test di valutazione della conoscenza della lingua Inglese fruibile *on line* (Figura 12).

Java è un linguaggio di programmazione assai articolato e versatile, e affrontare lo sviluppo ex novo di *applet* didattici richiede le competenze specifiche di un programmatore quanto meno a suo agio con le metodologie *object oriented*. Assai diversa invece è la situazione in cui si voglia utilizzare un *applet* già esistente, acquisito per via commerciale o come *free software*, integrandolo in una pagina Web di propria produzione; in questo caso l'operazione non richiede particolari competenze oltre a quelle necessarie per produrre pagine HTML, e si prevede per il prossimo futuro una notevole espansione del mercato delle componenti riusabili sviluppate in Java [Morrison 97].

Allo scopo di massimizzare le potenzialità di diffusione e riuso di componenti software

RISORSE JAVA *ON LINE*

Chi voglia approfondire la conoscenza di Java non ha che l'imbarazzo della scelta: il numero di siti Internet che ospitano materiale, esempi, tutoriali, strumenti è impressionante. Le informazioni in questa scheda sono pertanto un campione assai parziale della disponibilità, ma costituiscono comunque un buon punto di partenza.

La fonte di riferimento per Java e *JavaBeans* è senza dubbio il sito *JavaSoft*: http://www.javasoft.com/, che mette a disposizione documentazione, software, riferimenti ad altri siti e gruppi di interesse. Val la pena segnalare che il software *Java Development Kit* è (almeno fino ad oggi) completamente gratuito e può essere liberamente scaricato da questo sito. È anche possibile ricevere gratuitamente per posta elettronica la *Java Developer Connection(sm) Newsletter*, iscrivendosi alla URL http://developer.javasoft.com/servlet/RegisterServlet
Un'altra fonte che molti considerano essenziale è il sito *Gamelan*: http://java.developer.com/, una ricchissima raccolta di risorse suddivise per categorie, ognuna che rimanda a informazioni, codice sorgente, esempi, ecc. Ad esempio, la categoria *Educational* elenca oltre 1200 riferimenti ad *applet* didattici, suddivisi per sotto–categorie: Biologia, Chimica, Fisica, Ingegneria, Linguistica, Matematica, siti per bambini, ecc. (http://www.gamelan.com/pages/Gamelan.educational.html)
Per chi voglia approfondire gli aspetti tecnici della programmazione in Java è senz'altro da segnalare il sito http://www.eckelobjects.com/, curato da Bruce Eckel, che rende disponibile il suo libro *Thinking in Java*, liberamente scaricabile e stampabile. Il libro è un'ottima introduzione al linguaggio, molto curata anche dal punto di vista didattico.
Un altro ottimo tutoriale Java, curato da *Sun Microsystems*, è disponibile alla URL http://java.sun.com/Series/Tutorial/ e può essere scaricato in locale e consultato mediante un *browser* HTML (es. *Netscape*, *Internet Explorer*).
Tra i siti Italiani è indispensabile citare http://jis.rmnet.it/ che, interamente dedicato a Java, tra le altre informazioni elenca anche un buon numero di riferimenti a persone che in Italia programmano nel linguaggio; http://www.anfiteatro.it/java.html, curato da Fabio Ciucci, che ospita un corso in Italiano su Java; e *MokaByte*, la prima rivista *on line* italiana su Java, alla URL http://www.programmers.net/Riviste/moka/current/.

generiche scritte in Java è in via di sviluppo una particolare tecnologia, denominata *JavaBeans*, che persegue l'obiettivo di "scrivere (il software) una sola volta, eseguirlo su qualunque piattaforma e riusarlo ovunque". Rispetto ad un generico programma o *applet* Java, una componente *JavaBeans* deve rispettare una serie di regole formali e strutturali finalizzate a massimizzarne le possibilità di riuso. Il fatto che *JavaSoft* (gli "inventori di Java") e *Sun* si siano pesantemente impegnati su questo terreno ha ovvie motivazioni di mercato e può essere letto come una tendenza a contrastare il predominio di Microsoft nel campo delle tecnologie di *network programming* (v. più avanti la sezione dedicata ad *ActiveX*). Per ora *JavaBeans* è poco più che una convenzione normativa delle modalità di costruzione di componenti Java: non esiste al momento una vasta disponibilità di componenti pronte per l'uso, ne' sul mercato commerciale ne' su quello *public domain*, come invece esiste per le componenti *ActiveX*. Vari produttori di strumenti di sviluppo software (*Symantec*, *Borland*, IBM ed altri) hanno tuttavia investito risorse consistenti nella direzione di *JavaBeans*, e non è improbabile che nei prossimi mesi si assista ad una rapida affermazione di questa tecnologia; uno scenario in cui siano disponibili ricche librerie di componenti riusabili, specificamente sviluppate a scopi didattici e fruibili su un'ampia gamma di piattaforme, risulta invero particolarmente allettante a chi si propone di sviluppare pagine Web con finalità educative.

ActiveX[10]

La tecnologia *ActiveX* può per molti versi essere considerata simile a quella Java e *JavaBeans*, perché gli oggetti *ActiveX* sono componenti software (spesso chiamati "controlli") integrabili nelle pagine Web ed eseguiti dal *browser* del cliente, atti a consentire animazioni, multimedialità e interattività. La genesi di questa tecnologia è tuttavia assai diversa, e vale la pena ripercorrerla brevemente per capirne le sostanziali differenze rispetto a Java.
Il primo passo fu compiuto da Microsoft con la definizione, verso la fine degli anni 80, del modello OLE (*Object Linking and Embedding*), uno standard di rappresentazione dei documenti compositi che consente l'interoperabilità tra varie applicazioni facilitando l'integrazione di componenti eterogenee nei documenti. Così ad esempio è possibile importare in un documento Word una tabella

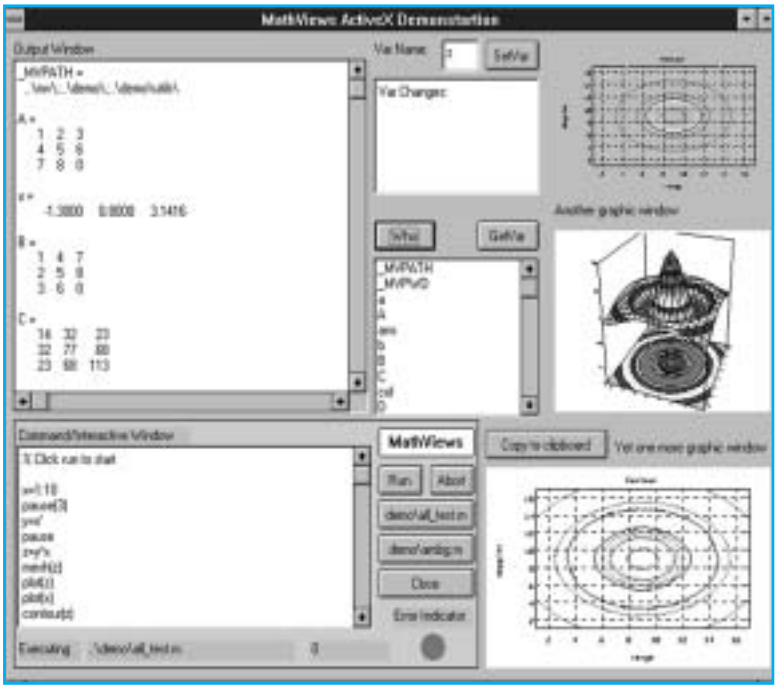


Figura 13
La demo del controllo ActiveX Mathplorer.

sviluppata con Excel, una presentazione prodotta con PowerPoint o un qualunque semilavorato ottenuto da uno strumento *OLE compliant* (che rispetti, cioè, lo standard OLE). L'evoluzione di OLE, OLE2, realizzò intorno al 1992 una più stretta integrazione tra le applicazioni e il sistema operativo Windows (consentendo ad es. il *drag and drop* di oggetti da un'applicazione ad un'altra), definì un modello di documento complesso, perfezionò il concetto di architettura *client–server* e gettò le basi per una futura architettura distribuita.

Nella prima metà del 1996 Microsoft introduce la tecnologia *ActiveX*, che come OLE è basata su COM (*Component Object Model*): questa specifica definisce uno standard di rappresentazione binaria degli oggetti indipendente dal linguaggio di programmazione, e consente a due applicazioni lo scambio di oggetti senza che l'una sia a conoscenza dei dettagli implementativi messi in atto dall'altra. *ActiveX* tuttavia si differenzia da OLE nei servizi offerti alle applicazioni: OLE usava COM per consentire la creazione di documenti complessi, composti da oggetti di varia natura; *ActiveX* è invece intesa a fornire un'infrastruttura agile e flessibile per inserire controlli nelle pagine Web e gestire appropriatamente l'interazione con l'utente. Il modello COM viene esteso a DCOM (*Distributed* COM) per consentire a processi remoti di condividere oggetti e funzioni attraverso la mediazione della connessione telematica. L'architettura è strettamente integrata con il sistema operativo, per cui quando un controllo *ActiveX* viene utilizzato per la prima volta su un computer cliente, esso viene automaticamente registrato nel sistema locale per velocizzare successivi accessi. Anche qui grande attenzione è posta alla sicurezza dei dati (assenza di virus, riservatezza dei dati trasferiti) per rispondere alle necessità di consistenti porzioni del mercato della telematica (banche, commercio via Internet, ecc.). È evidente inoltre il vantaggio conseguito rispetto alla tradizionale architettura Netscape, basata su estensioni delle funzionalità multimediali del browser (i *plug-in*) che devono essere preventivamente installati con un espresso intervento dell'utente: i controlli *ActiveX* si installano da soli e solo se ce n'è bisogno.

I controlli *ActiveX* consentono di conferire alle pagine Web notevoli funzionalità interattive. Ad esempio, in Figura 13 è rappresentata la pagina dimostrativa di *MathXplorer*, un controllo *ActiveX* sviluppato da *The MathWizards*[11] che tratta argomenti relativi all'algebra lineare, al calcolo matriciale e all'elaborazione di segnali digitali. Lo studente inserisce espressioni algebriche e matriciali (nella *Interactive Window*, in basso a sinistra); quando il bottone *Run* è premuto esse vengono valutate, e i risultati sono visualizzati in forma simbolica (nella *Output Window*, in alto a sinistra) o grafica (nella *Graph Window*, a destra).

Grazie ai controlli *ActiveX* le pagine Web acquistano spesso una dimensione ulteriore di potenzialità interattive e multimediali. Ad esempio il controllo *RealPlayer*, prodotto da *Progressive Networks, Inc.*[12], provvede alla presentazione di segmenti video o audio di buona qualità sfruttando un'innovativa tecnologia *stream*[13]: la Figura 14 cattura un controllo *RealPlayer* durante la presentazione di un filmato relativo ad un'intervista a Gorbachev[14].

Figura 14
Il controllo ActiveX RealPlayer.

Per avere un'idea delle potenzialità in campo didattico di *ActiveX* si consideri *Microsoft Agent*[15], un controllo recentemente reso disponibile da Microsoft; esso consente di aggregare ad una pagina Web un "personaggio" che, rappresentato da un'animazione *esterna* alla finestra del *browser*, si muove, parla (in sintesi vocale) ed è persino in grado di accettare comandi impartiti a voce dall'utente (purché, ovviamente, il sistema sia dotato di un microfono e dell'opportuno modulo software per il riconoscimento della voce). Il personaggio può essere usato in ruoli interattivi ed amichevoli per fornire aiuto contestualizzato all'utente, per stimolarlo con domande e suggerimenti, o ancora per rinforzare particolari metafore (ad es. l'anfitrione che ci conduce nella visita di un museo virtuale, o lo scienziato che ci aiuta a risolvere un problema di fisica…). Le potenzialità d'uso di strumenti di questo tipo in campo didattico sono numerose, e includono sia ambienti tutoriali sia interventi didattici basati su strategie meno direttive, dove l'esigenza di mantenere alti i livelli di motivazione ed attenzione dello studente è particolarmente sentita. La Figura 15 mostra un esempio dell'uso di un agente, raffigurato da un robot (*Robby*).

Esistono controlli *ActiveX* che consentono di ospitare nella pagina Web una presentazione *PowerPoint*, con controllo locale della sequenza di presentazione delle diapositive. Grazie ad altri controlli *ActiveX* è possibile inserire nelle pagine Web tabelle direttamente connesse a database esterni. Ci limitiamo qui a segnalare come fonti il sito Microsoft[16], quello CNET[17] e il già citato *Gamelan*[18], tutti assai ricchi sia di materiale direttamente scaricabile, sia di riferimenti a documentazione, esempi, gruppi d'interesse, ecc.

A differenza di Java, l'architettura *ActiveX* non è indipendente dalla piattaforma hardware e software: si tratta di una soluzione proprietaria di Microsoft che richiede l'utilizzo come *browser* di *Internet Explorer* (versione 3.0 o successive). È questo l'unico vero limite di *ActiveX*: le pagine che contengono un controllo *ActiveX* non verranno correttamente visualizzate da un *browser Netscape*[19]. Come vedremo più avanti, è possibile ovviare a questi inconvenienti, a prezzo di uno sforzo di programmazione maggiore.

I controlli *ActiveX* possono essere sviluppati in vari linguaggi: C++, Visual Basic, Java, *Borland Delphi*. Microsoft rende disponibile

Figura 15
Una pagina Web con un Microsoft Agent (Robby).

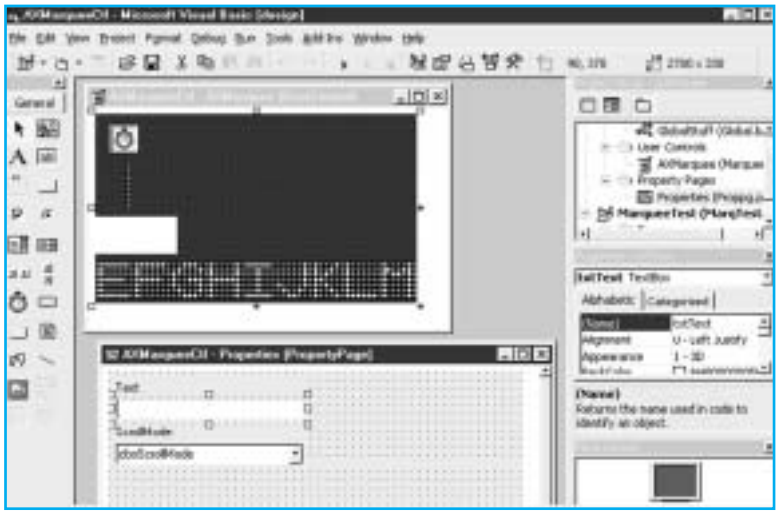


Figura 16
L'ambiente Microsoft Visual Basic Control Edition.

Figura 17
Il test del controllo ActiveX Marquee.

gratuitamente *Visual Basic Control Edition*, un ambiente facilitato per lo sviluppo di semplici controlli *ActiveX*; in Figura 16 è rappresentato il programma in questione, nel quale è stato caricato come esempio il controllo "*Marquee*" che consente la visualizzazione di una stringa di caratteri in movimento. Il test del controllo può essere effettuato direttamente dall'interno dell'ambiente autore. Livelli maggiori di complessità richiedono di solito le competenze di un programmatore professionista; anche qui è invece estremamente semplice riusare controlli sviluppati da altri: sono numerosissimi i prodotti

ActiveX distribuiti via Internet, spesso gratuitamente.

Scripting

Con il termine *scripting* si intende la possibilità di definire componenti software eseguibili che arricchiscono le funzionalità offerte da varie applicazioni ed ambienti. Una macro Word o Excel, ad esempio, è uno *script*; anche ambienti ipertestuali come Toolbook o Hypercard possono essere programmati con opportuni *script*. Le pagine WWW si basano su un linguaggio, HTML, che nella sua forma originale consente soltanto di specificare, per ogni componente del testo, il suo ruolo strutturale nell'architettura del documento ipertestuale: un paragrafo o una stringa del testo possono facilmente essere "marcati" come titolo, o come citazione, o ancora come riferimento ipertestuale: ne consegue la definizione automatica e portabile di alcune caratteristiche di presentazione di quella componente. HTML è un linguaggio puramente descrittivo, nel quale non è possibile specificare un comportamenti attivo che non sia stato a priori previsto da chi ha disegnato il linguaggio e associato ad una determinata *tag*. Ad esempio, una pagina puramente HTML non può "decidere" modi di presentazione alternativi in base al contesto in cui viene fruita: HTML non ammette strutture di controllo "IF…THEN". Per ovviare a questa limitazione i più diffusi *browser* sono stati dotati della capacità di interpretare uno o più *scripting languages*, linguaggi imperativi in cui è facile definire comportamenti attivi.

In passato l'esigenza di inserire nelle pagine Web funzionalità computazionali non gestite dal *browser* veniva affrontata con il meccanismo CGI (*Common Gateway Interface*); che consente di arricchire il *server* Web con servizi accessibili dal *browser* cliente attraverso il protocollo HTTP. Questa modalità richiede tuttavia una transizione che investe il canale telematico, e presenta spesso tempi di risposta che dipendono dal traffico di rete. L'uso degli *script*, invece, permette di risolvere la maggior parte dei problemi localmente al *browser*, senza coinvolgere il *server* e la rete.

A differenza dei più titolati linguaggi di programmazione (Java, C++) gli *scripting languages* non richiedono al programmatore competenze informatiche spinte e si rivolgono pertanto ad una base d'utenza assai ampia, offrendo sintassi facilitate, funzionalità specializzate e una curva d'apprendimento assai morbida. Per motivi di sicurezza, in genere gli *scripting languages* non possono scrivere file sul disco né accedere direttamente alle zone di memoria, rendendo così virtualmente impossibile la produzione di virus ed altre analoghe nefandezze.

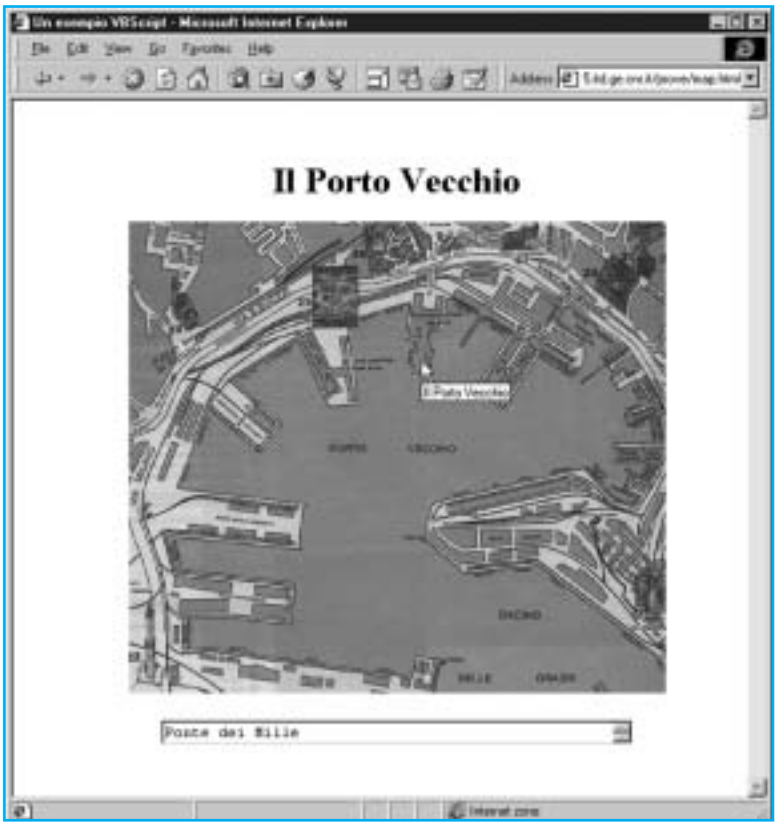


Figura 18
Una mappa "cliccabile" realizzata con VR-Script.

VBScript

VBScript è uno *scripting language* basato su Visual Basic; secondo una stima Microsoft[20], nel mondo operano circa tre milioni di programmatori Visual Basic: chi sa già programmare in questo linguaggio non avrà alcuna difficoltà ad usare *VBScript*. L'interprete di *VBScript* è integrato nel *browser* Internet Explorer a partire dalla versione 3.0: quando una pagina Web contenente codice *VBScript* viene caricata nel browser, l'interprete viene automaticamente attivato e lo script eseguito.

VBScript consente di integrare e gestire nelle pagine Web vari tipi di oggetti: alcuni sono messi a disposizione dall'interprete del linguaggio, altri da Internet Explorer; ad esempio, è facile predisporre uno *script* che inserisca nella pagina la frase "Buon Giorno!" se

l'orologio di macchina segnala un'ora mattutina, oppure "Buona Sera!" in ore pomeridiane. L'esempio è banale, ma rende ragione della flessibilità introdotta da uno *scripting language* rispetto alla staticità delle pagine puramente HTML. La Figura 18 mostra una mappa dove grazie a *VBScript* sono state definite alcune zone "calde": quando il mouse transita su una di queste zone, nel campo di testo sottostante la mappa appare una descrizione del particolare; ad ogni zona corrisponde una specifica *ancora* ipertestuale che definisce l'indirizzo cui il *browser* salta quando l'utente "clicca" in quella zona.

Pur senza scendere nei dettagli informatici più specialistici, può essere utile esaminare alcuni frammenti del codice relativo alla mappa di Figura 18. In Figura 19 si può vedere come la *tag* HTML **<SCRIPT>** sia usata per circoscrivere la definizione della procedura **Immagine_onMouseMove()**, che viene automaticamente attivata quando il mouse si sposta sull'oggetto **Immagine**. La procedura assegna al campo **Descrizione** la stringa opportuna, in dipendenza della posizione del mouse.

Altri oggetti manipolabili da *VBScript* sono i controlli *ActiveX* inseriti nella pagina Web dall'autore. In effetti sarebbe arduo integrare le componenti *ActiveX* nel codice HTML, senza la funzione di "collante" svolta da uno *scripting language*.

Apparentemente l'uso di *VBScript* richiede qualche competenza informatica in più rispetto a quelle, minimali, necessarie a chi voglia realizzare pagine Web in semplice HTML. Esistono tuttavia strumenti che, offrendo un'interfaccia quasi completamente


```
<SCRIPT language=vbscript>
        Sub Immagine_onMouseMove()
            y = window.event.offsetY
            x = window.event.offsetX
            If InRect(x, y, 44, 236, 155, 292) Then
                Descrizione.Value = "Ponte Biagio Assereto"
                ElseIf InRect(x, y, 40, 309, 149, 360) Then
                        …
        End Sub
</SCRIPT>
<IMG id="Immagine" alt="Il Porto Vecchio" src="porto.gif" …>
<TEXTAREA name="Descrizione" rows=1 cols=50></TEXTAREA>
```

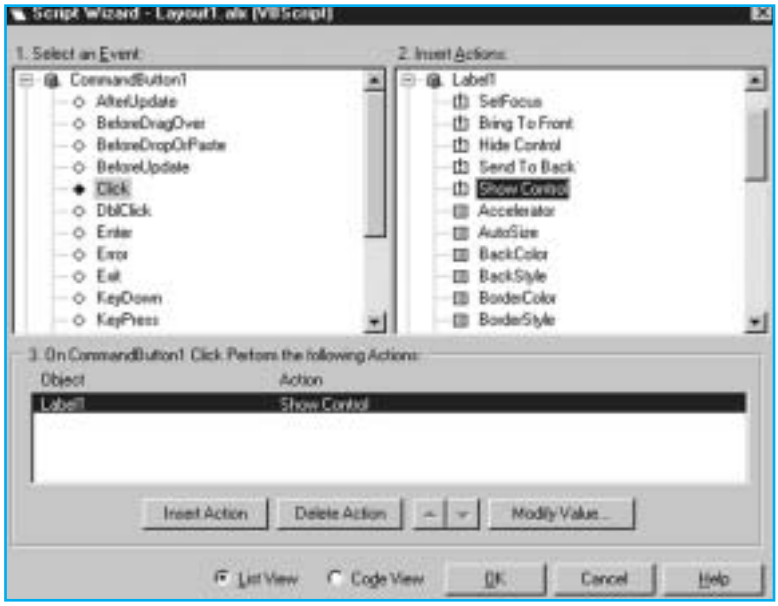
Figura 19
Un frammento del codice VB-Script.

visuale, facilitano il compito dell'autore, isolandolo per quanto possibile dalle idiosincrasie sintattiche del linguaggio. Uno di questi è *Microsoft ActiveX Control Pad*, liberamente scaricabile via rete[21], che consente di produrre pagine Web con bottoni, elementi di dialogo, controlli *ActiveX*, menu ecc. personalizzabili e genera automaticamente il codice *VBScript* necessario a connettere il tutto. L'autore interagisce con l'ambiente in una modalità molto simile a quella tipica di un editore grafico: gli oggetti vengono selezionati su una *pallette* iconica, trascinati sulla pagina e personalizzati modificando i valori di una tabella di proprietà (Figura 20).

Figura 20
Uso di *Microsoft ActiveX Control Pad*: definizione del *layout*.

Lo script viene anch'esso definito selezionando gli eventi significativi e le azioni relative: in Figura 21 è stato selezionato come evento il "clic" sull'oggetto **CommandButton1** (il bottone "Premi qui" di Figura 20) e come azione conseguente la visualizzazione del campo di testo verde **Label1** ("Ciao" in

Figura 21
Uso di *Microsoft ActiveX Control Pad*: definizione dello script.



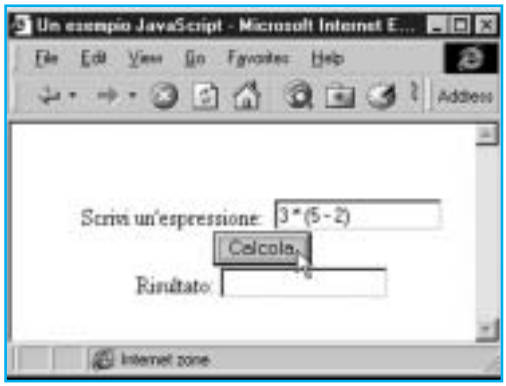


Figura 22
Un semplice calcolatore JavaScript.

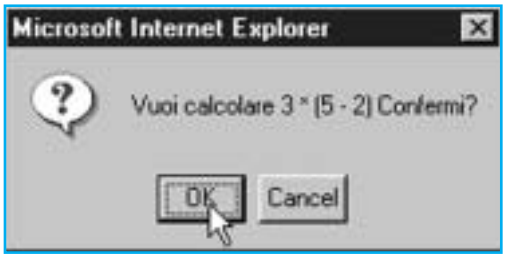


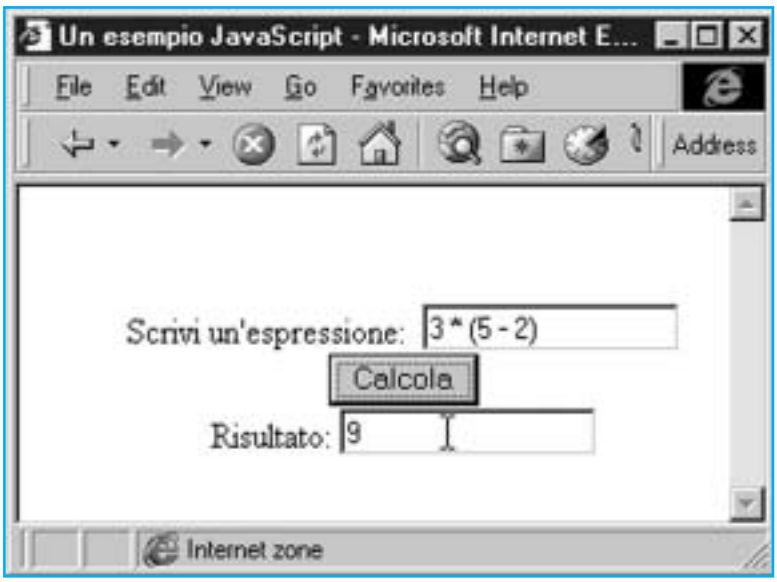
Figura 23
Un alert box.

Figura 20), prima dichiarato invisibile. Ciò è sufficiente a che *Microsoft ActiveX Control Pad* inserisca automaticamente il codice *VBScript* nella pagina Web in produzione. Anche chi non desidera approfondire i tecnicismi del linguaggio può così cimentarsi nella realizzazione di semplici *script*, grazie alla curva di apprendimento particolarmente "morbida" consentita da questo tipo di ambienti.

JavaScript

Nonostante il nome, sotto molti aspetti *JavaScript* è cosa assai diversa da Java, e gli aspetti in comune sono prevalentemente circoscritti alla sintassi. *JavaScript* è uno *scripting language*, come *VBScript*, le cui istruzioni vengono inserite nel codice HTML delle pagine Web e sono direttamente interpretate dal *browser*[22] al momento della fruizione, mentre Java è (come già visto) un linguaggio con cui possono anche essere realizzate applicazioni *stand alone*. Dal punto di vista funzionale, *JavaScript* è assai simile a *VBScript*: può essere usato per gestire l'interazione con l'utente con modalità più sofisticate di quelle offerte dal semplice HTML. Per esempio, una funzione *JavaScript* può verificare la validità formale di dati digitati dall'utente (numeri telefonici, codici postali, ecc.); la verifica avviene localmente al *browser* del cliente, senza coinvolgere il server in una transizione via rete, migliorando quindi i tempi di risposta e diminuendo il traffico di rete. *JavaScript* può anche controllare la presentazione di oggetti multimediali rispondendo non solo ad azioni dell'utente, ma anche ad eventi generati internamente dal sistema (ad esempio un *timeout*).

Figura 24
Il valore dell'espressione calcolata viene visualizzato in un campo di testo.



Da quanto visto, *JavaScript* e *VBScript* sono grosso modo equivalenti. Al di là delle simpatie che ogni programmatore può eventualmente nutrire, la differenza fondamentale è che mentre *Internet Explorer* è in grado di interpretare sia *JavaScript* che *VBScript*, Netscape conosce invece solo *JavaScript* e, salvo l'adozione di specifici *plug-in*, è "sordo" a *VBScript*.

La Figura 22 fornisce un semplice esempio di come *JavaScript* possa arricchire le potenzialità di calcolo di una pagina Web. Dopo aver digitato un'espressione algebrica nel campo di input in alto, l'utente preme il pulsante *Calcola*. Lo script chiede conferma attraverso l'attivazione di un'*alert box* (Figura 23) e, ottenutala, visualizza il risultato del calcolo nel campo di testo in basso (Figura 24).

In Figura 25 è mostrato il codice relativo a questo esempio. Un aspetto significativo del codice è la possibilità, nella definizione di una funzione, di far riferimento agli elementi della pagina HTML: il campo di input si chiama **form.espressione.value**, quello di output si chiama **form.risultato.value**. Analogamente nel bottone l'attributo **ONCLICK** riferisce la funzione **calcola (this.form)**, che verrà automaticamente invocata quando l'utente premerà il bottone.

Chi volesse approfondire la conoscenza di

```
<SCRIPT LANGUAGE="JavaScript">
 function calcola(form) {
   if (confirm("Vuoi calcolare " + form.espressione.value + " Confermi?"))
     form.risultato.value = eval(form.espressione.value)
   else
     alert("Come vuoi.")
}
</SCRIPT>
…
Scrivi un'espressione:
<INPUT TYPE="text" NAME="espressione" SIZE=15><BR>
<INPUT TYPE="button" VALUE="Calcola" ONCLICK="calcola(this.form)"><BR>
Risultato:
<INPUT TYPE="text" NAME="risultato" SIZE=15 >
```

Figura 25
Il codice JavaScript dell'esempio in figura 22.

JavaScript può consultare un buon tutoriale[23] e la documentazione di riferimento[24] sul sito di Netscape. Una serie di articoli introduttivi [Kahn 97] è disponibile sul sito CNET[25]. C'è anche su un sito tutto dedicato[26] a *JavaScript*, con una collezione di oltre 500 esempi tutti liberamente scaricabili.

Active Server Pages

Finora abbiamo visto come uno *scripting language* possa migliorare sensibilmente le funzionalità disponibili in un *browser* Web. È tuttavia possibile usare gli *script* anche dal lato *server*, per aggiungere flessibilità senza tuttavia affrontare le complesse problematiche delle interfacce ISAPI e NSAPI[27]. Il *server* HTML prodotto da Microsoft per la sua piattaforma Windows NT[28] offre infatti un ambiente di esecuzione (*Active Server Pages*, abbreviato ASP) in grado di interpretare codice *VBScript* inserito nelle pagine HTML prima che queste vengano trasferite al *browser* che ne ha fatto richiesta.
In pratica, è possibile inserire uno *script* che costruisce dinamicamente la pagina sul *server*, ad esempio per inserirvi dati freschi estratti da un database; inoltre, poiché la richiesta di una pagina perviene al *server* corredata delle informazioni relative al *browser* che l'ha generata, lo *script* può riconoscere il tipo di *browser* del cliente, e preparare la pagina HTML sul momento, inserendo soltanto quelle componenti che il *browser* è in grado di visualizzare. Una delle maggiori difficoltà nel predisporre pagine Web, infatti, è causata dalla grande variabilità di funzioni disponibili sui *browser* usati dai potenziali fruitori: oltre ai già citati *Netscape* e *Internet Explorer* vi sono in giro numerose altre applicazioni più o meno equivalenti quando si tratta di gestire codice HTML "classico", ma incapaci di interpretare correttamente le recenti estensioni HTML, gli *scripting language*, i controlli *ActiveX* o gli *applet* Java. Gli stessi *Netscape* e *Internet Explorer* sono presenti in varie versioni, tutte sottilmente incompatibili tra loro. Lo *script* ASP può pertanto determinare in anticipo, quando gli

```
<% Set bc = Server.CreateObject("MSWC.BrowserType") %>
<% If (bc.Browser = "IE") Then %>
        <object id="finestra" classid="CLSID:A23D7C20-CABA-11CF-A5D4-00AA00A47DD2"
          codebase="http://activex.microsoft.com/controls/iexplorer/iepopwnd.ocx">
        </object>
        <SCRIPT Language="VBSCRIPT">
          Sub fiore_onMouseOver()
                finestra.Popup "http://ls-p5/prove/tip.html", False
          End Sub
          Sub fiore_onMouseOut()
                finestra.dismiss
          End Sub
          </SCRIPT>
<% End If %>
<IMG id=fiore src="rose.gif">
```

Figura 26
Un frammento di pagina ASP.

Figura 27
Un tip con testo in formato HTML

perviene la richiesta, le capacità del *browser* cliente, e costruire di conseguenza la pagina richiesta.
La Figura 26 mostra un frammento di codice dove è usato sia uno script ASP (sul lato *server*) che uno script *VBScript* (sul lato cliente). La pagina HTML contiene soltanto una componente visibile, una figura: **rose.gif**. Le istruzioni delimitate da **<%** e **%>** vengono interpretate sul *server*: se il *browser* che ha richiesto al pagina in questione è *Internet Explorer* nella pagina stessa viene inserito il codice *VBScript* compreso tra **<% If …%>** ed **<% End If %>**; in caso contrario al *browser* perverrà solo la figura.
Il codice *VBScript* dichiara l'uso un controllo *ActiveX* (**iepopwnd.ocx**), ne definisce l'identificativo e l'indirizzo, ed associa all'evento generato dal transito del mouse sulla figura (**onMouseOver**) la chiamata del metodo **Popup**, che visualizza un testo formattato HTML in una finestra *pop–up* (Figura 27). Quando il mouse viene spostato fuori dalla figura (**onMouseOut**) la finestra *pop–up* si chiude.
Un testo molto approfondito sulla tecnologia ASP[29] [Cornig et al. 97] è accessibile in rete.

Cascade Style Sheets

I fogli di stile (*style sheets*) possono essere applicati a pagine HTML per controllare lo stile di presentazione del documento: la scelta dei colori e dei font, le dimensioni dei margini, ecc. Gli stili possono essere definiti all'interno dello stesso file HTML, oppure in un file separato, per essere facilmente applicati a più pagine. Ogni elemento HTML può trarre le sue specifiche di stile da più sorgenti, che si mescolano in "cascata": le regole specificate in un foglio hanno la precedenza su quelle dei fogli che lo precedono. Le specifiche di stile possono essere applicate all'intera pagina, o solo ad alcuni suoi elementi.
È facile in questo modo modificare il comportamento standard del proprio *browser*: si può ad esempio specificare nel foglio di stile che tutti i paragrafi (marcati **<P>** in HTML) dovranno essere presentati in colore blu e con un font molto grande:

```
<STYLE>
  P {COLOR = blue; FONT-SIZE: xx-large}
</STYLE>
```

Una funzionalità del genere può essere molto utile per ovviare a problemi di lettura, o per proiettare a grande schermo con un uditorio numeroso.
Un'altra funzionalità interessante (ma per ora presente solo a livello di specifiche, e non ancora realizzata nei *browser* correnti) è costituita dalla possibilità di specificare stili alternativi per vari *media* di presentazione: ad esempio è possibile dichiarare che in presenza di un dispositivo che converte il testo in un parlato audio[30] le ancore ipertestuali siano evidenziate da segnali sonori:

```
<STYLE MEDIA="aural">
  A {CUE-BEFORE = url(bell.aiff); CUE-AFTER: url(dong.wav)}
</STYLE>
```

In questo modo la forma della presentazione può essere adattata allo specifico *medium* usato per la sua fruizione: stampante, proiettore, dispositivo tattile Braille, ecc.
In effetti queste estensioni snaturano un po' l'intento originale dello HTML che, come linguaggio di *marcatura*, dovrebbe definire la struttura e il ruolo delle componenti del documento, non direttamente il loro formato e aspetto di presentazione. Tuttavia il livello di espressività sopra esemplificato offre a chi predispone pagine Web le potenzialità di un editore grafico tradizionale, e non a caso i *word processor* più diffusi prestano sempre

Figura 28
Al transito del mouse il paragrafo cambia colore.

Figura 29
Un "click" sul paragrafo lo espande e visualizza del testo prima nascosto.

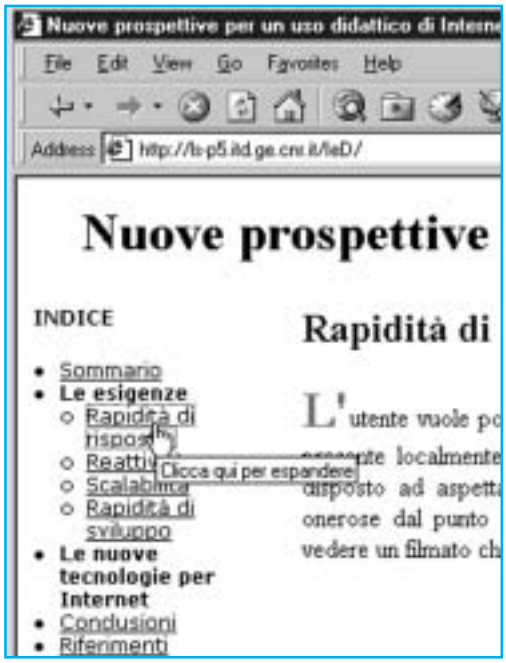
Figura 30
Un ulteriore livello di espansione.

maggior attenzione allo HTML. Se fino ieri *MSWord* poteva produrre codice HTML solo grazie ad una speciale estensione, e con non poche grane aggiuntive per l'operatore, l'attuale versione di punta del prodotto consente la produzione di documenti ipermediali navigabili da un *browser* HTML e offre una serie di funzionalità specificamente orientate all'*editing* di pagine Web. Forse domani HTML sarà il formato standard per la rappresentazione e l'interscambio di documenti.

Dynamic HTML

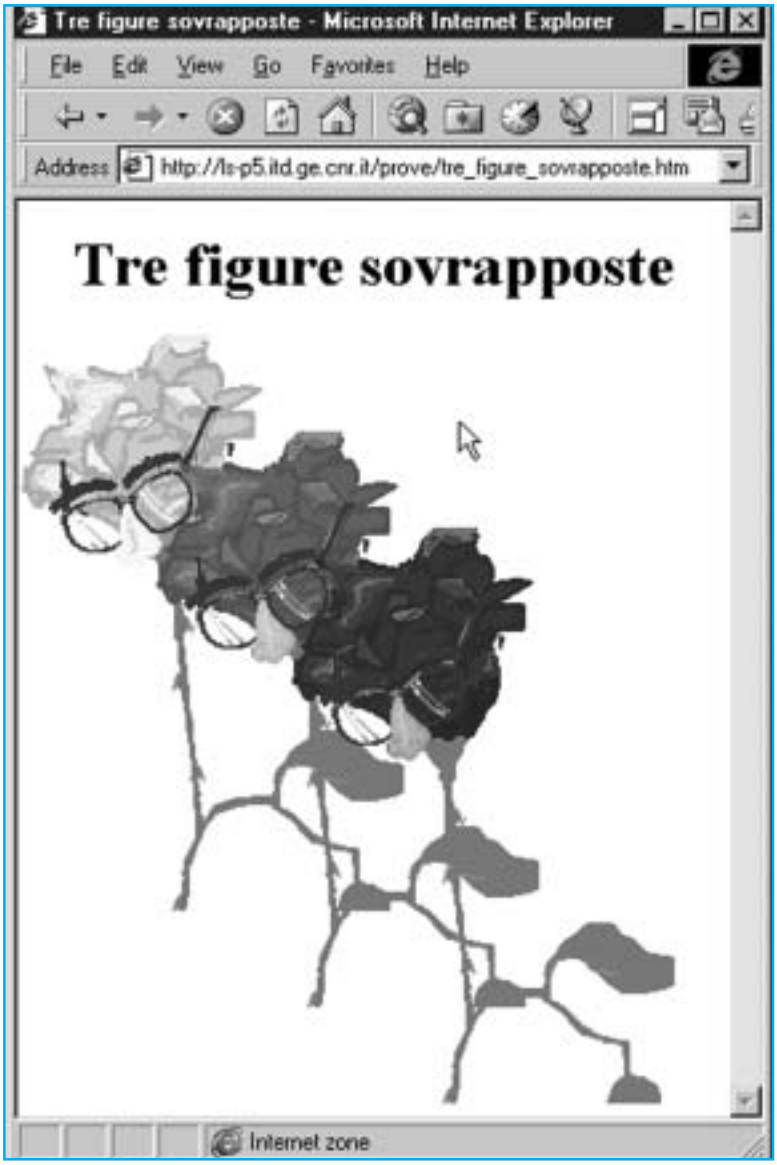
Dynamic HTML (DHTML), un'altra funzionalità recentemente introdotta nello standard HTML, apre nuove prospettive alla creazione di pagine Web interattive. DHTML definisce un modello per la gestione delle componenti del documento (in gergo, un *object model*) che consente agli *script* di manipolare gli stili e gli attributi degli elementi nella pagina (gli *oggetti*) e perfino di rimpiazzare elementi esistenti con altri, nuovi. DHTML permette agli sviluppatori di sbizzarrire la propria creatività, migliora il controllo della presentazione e spesso abbrevia i tempi di visualizzazione della pagina, perché riporta molti comportamenti dinamici al lato *browser*. In DHTML una data porzione di testo può assumere dimensioni, colore, font o stile diversi quando il mouse vi transita sopra (Figura 28); è possibile realizzare menu gerarchici le cui voci si espandono o si contrag-

```
<style>
  .fiore1 {position:absolute; left:0px;
           top:70px; z-index:1}
  .fiore2 {position:absolute; left:100px;
           top:120 px; z-index:2}
  .fiore3 {position:absolute; left: 200px;
           top:170 px; z-index:3}
</style>
…
<p><img src="fiore1.gif" class="fiore1">
<p><img src="fiore2.gif" class="fiore2">
<p><img src="fiore3.gif" class="fiore3">
```



gono, evidenziando o nascondendo i livelli di maggior dettaglio, a richiesta dell'utente (Figura 29, Figura 30); le componenti della pagina possono essere posizionate a piacere (Figura 31); gli elementi di una tabella possono assumere valori provenienti da un database locale e possono essere riordinati e filtrati secondo le esigenze dell'utente. Tutte queste funzionalità sono realizzate localmente al *browser*, e non richiedono una transizione telematica che coinvolga il *server*.
Il posizionamento delle componenti nella pagina è un'interessante possibilità offerta dal DHTML. Di ogni elemento (testuale o grafico) è possibile definire la posizione in termini assoluti (sulla pagina) o relativi (ad altri elementi); si può anche controllare l'ordinamento sul cosiddetto "asse Z", ortogonale alla pagina, per specificare che un elemento deve sovrapporsi parzialmente ad un altro (Figura 31). Per tutto quanto riguarda DHTML il riferimento è il sito della Microsoft[31] che mette a disposizione testi introduttivi, tutoriali, esempi e strumenti dedicati a questa particolare tecnica.
Al momento solo le più recenti versioni dei browser Netscape e Internet Explorer sono in grado di gestire DHTML, oltretutto in modalità mutuamente incompatibili. Le previsioni sono tuttavia incoraggianti: sia perché i browser di ultima generazione acquisiranno sempre maggiore diffusione, sia perché molti produttori software sono impegnati nel fornire strumenti autore e semilavorati basati su DHTML, con la conseguente costituzione di un vero e proprio mercato per questa tecnologia [Paul 97].

Figura 31
Il posizionamento di elementi sulla pagina.



Conclusioni
L'articolo ha presentato alcune tecniche innovative nel campo della produzione di pagine Web, con particolare riferimento alla loro potenzialità in campo didattico. L'enumerazione di tali tecniche non ha alcuna ambizione di completezza: sono stati trascurati alcuni prodotti e soluzioni che meriterebbero di essere esplorati a fondo, anche se la loro applicabilità in contesto educativo appare per il momento ridotta: *push technology*, strumenti collaborativi basati su Internet (ad esempio *Netmeeting*, *FirstClass*, ecc.)
Cercare di presentare su carta esempi di comportamenti dinamici è un'impresa assai ardua, che costringe chi scrive ad inevitabili inesattezze, e coinvolge il lettore in sforzi di fantasia ed astrazione non indifferenti. Per ovviare a questi inconvenienti è stata realizzata una versione HTML di questo articolo, navigabile in Internet[32], che usa realmente le tecniche e gli strumenti oggetto della trattazione e ne offre numerosi esempi, arricchiti di tutte le informazioni (sorgenti completi e commentati, ulteriori riferimenti, ecc.) che nell'edizione a stampa non hanno trovato posto.

Note

1 http://www.alchera.it
2 http://www.mapquest.com
3 http://www.iflyswa.com
4 http://www.pathfinder.com
5 http://www.ida.liu.se/~her/npp/demo.html
6 http://jersey.uoregon.edu/vlab
7 http://jersey.uoregon.edu/vlab/Voltage/index.html
8 http://biomedia.bio.purdue.edu/PerfectMatch/
9 http://web.tin.it/AIBSI/
10 *ActiveX* è in marchio registrato di Microsoft Corporation.
11 http://www.mathwizards.com/
12 http://www.real.com/ - di questo sito esiste una versione in Italiano: http://christie.prognet.com/international/it/
13 Il protocollo tradizionale HTTP prevede che una componente multimediale debba essere completamente scaricata sul computer di fruizione prima di poter essere presentata; la tecnologia *stream*, grazie ad un protocollo di trasferimento particolare e ad un sistema di memorizzazione intermedia, consente invece di iniziare la presentazione dell'oggetto multimediale quando la sua ricezione è ancora in atto, riducendo significativamente i tempi di attesa dell'utente.
14 http://www.real.com/products/realvideo/demo/index.ht ml
15 http://www.microsoft.com/workshop/prog/agent/
16 http://www.microsoft.com/activex/gallery/
17 http://activex.microsoft.com/
18 http://java.developer.com/
19 In realtà esistono *plugs–in Netscape* in grado di integrare controlli *ActiveX* (vedi ad esempio lo *ScriptActive plug-in* della *NCompass Labs* al sito http://www.ncompasslabs.com/scriptactive/index.htm), ma si tratta per lo più di soluzioni fornite da terze parti, la cui applicabilità presenta qualche limite.
20 http://www.microsoft.com/vbscript/us/techinfo/vbsfaq.htm#WhatIs
21 http://www.microsoft.com/workshop/author/cpad/
22 In effetti *JavaScript* può anche essere eseguito sul lato server, analogamente a *VBScript* nell'ambito ASP (vedi nel seguito)
23 http://home.netscape.com/eng/mozilla/Gold/handbook/javascript/
24 http://home.netscape.com/eng/mozilla/Gold/handbook/docs/atoz.html
25 http://www.cnet.com/Content/Builder/Voices/Kahn/091 097/?axd
26 http://www.javascripts.com
27 SAPI (*Internet Server Application Program Interface*) è un'API (cioè una libreria di funzioni) predisposta per la costituzione di moduli software di servizio eseguibili da un *server* HTML Microsoft in ambito Windows NT (vedi nota 28). NSAPI (*Netscape Server Application Program Interface*) è l'equivalente per i *server* Netscape. Collettivamente, questa classe di servizi prende il nome di *Fast CGI*.
28 IIS - *Internet Information Server* 4.0
29 http://www.mcp.com/que/asp/book/
30 Ancora poco diffusi da noi, dispositivi di sintesi sonora del testo cominciano ad essere usati negli USA per consentire a persone non vedenti di interagire con un elaboratore.
31 http://www.microsoft.com/workshop/author/dhtml
32 http://ls-p5.itd.ge.cnr.it/IeD

L'autore ringrazia Paolo Bianchetti, Giuseppe Damerio, Fabio Ferrari, Stefania Manca, Giorgio Olimpo, Donatella Persico e Mauro Tavella per aver accuratamente rivisto questo articolo. Desidera inoltre rivolgere un ringraziamento particolare ad Arrigo Frisiani, i cui consigli hanno significativamente migliorato questo lavoro. Di errori ed omissioni, l'autore rimane l'unico responsabile.

Una versione condensata di questo articolo è stata proposta per la pubblicazione su *Rivista di Informatica.*

Riferimenti Bibliografici

Barksdale J. "The browser is just the beginning", *Netscape Columns*, November 20, 1996 http://home4.netscape.com/comprod/columns/mainthing/strategy.html

Brethour P. "Internet Driving High-tech, report says", *The Globe and Mail*, February 26, 1997 http://www.holdroyd.com/news_release.html

Broersma M., "Study finds Net growth slowing", *Developer.com News*, EarthWeb Inc., December 23, 1997 http://www.developer.com/news/stories/archive/1297/122497_slowgrow.html

Cornig M., Elfanbaum S, Melnick D. "Working with Active Server Pages", Que Corporation, 1997 ISBN: 0-7897-1115.x http://www.mcp.com/que/asp/book/

Gates B. "Internet Strategy Workshop Keynote", December 7, 1995 http://www.internetvalley.com/clrk.html

Kahn C. "Object Oriented", *SuperScripter*, October 19, 1997 http://www.cnet.com/Content/Builder/Voices/Kahn/091097/?axd

Livraghi G. "Marketing in rete: è il momento di una rivoluzione copernicana", Convegno "Internet, il futuro in Rete", Centro Congressi di Assago Milanofiori, 12 marzo 1998 http://www.gandalf.it/nm/somedi00.htm

Mandò M. "Un italiano su venti adopera Internet", *La Repubblica.it*, 22 Settembre 1997

Mandò M. "Internet, ma quanti sono i naviganti italiani?", *La Repubblica.it*, 14 Maggio 1998

Morrison M "Presenting JavaBeans", Sams.net Publishing, Indianapolis (IN), ISBN 1-57521-287-0, 1997

Palmer M. "Building successful Web Apps", *WEB APPS*, SIGS Publications, NY, Jan/Feb 1997 http://www.sigs.com/publications/docs/oc/9701/oc9701.palmer.html

Paul F., "What to expect in 1998", *Builder.com*, CNET Inc., December 26, 1997 http://www.cnet.com/Content/Builder/Business/Paul/122497/?axd

Sun Microsystems "Java Developer Connection(sm) Newsletter", Vol. 1, N. 1, September 8, 1997 http://java.sun.com/events/jibe/partnerlist.html