54.1 URI e IRI

Un URI (*Uniform resource identifier*) è un indirizzo espresso attraverso una stringa di caratteri per identificare una risorsa fisica o astratta. La risorsa in questione è un'entità e la sua collocazione non si trova necessariamente all'interno di una rete. In pratica, il concetto di URI incorpora i concetti di URL (*Uniform resource locator*) e di URN (*Uniform resource name*).

Un URL identifica una risorsa rappresentando il metodo di accesso a questa; un URN identifica la risorsa attraverso un nome, il quale deve essere unico a livello globale e deve persistere anche quando la risorsa cessa di esistere o diventa inaccessibile.

Un IRI (*Internationalized resource identifier*) è un URL che consente però l'uso dei caratteri previsti da Unicode, ovvero dalla codifica universale.

54.1.1 Trascrivibilità

L'esigenza primaria degli indirizzi URI e degli IRI è la loro «trascrivibilità». Con questo termine si vuole fare riferimento alla facilità con la quale questi devono poter essere trascritti, sia a livello meccanico, sia a livello umano. In pratica:

- un URI o un IRI è composto da una sequenza di «caratteri» e non necessariamente da ottetti (byte);
- un URI o un IRI deve poter essere trascritto attraverso qualunque mezzo, come una pubblicazione stampata o un appunto fatto a mano, in tal senso non può utilizzare caratteri particolari che possono mancare in un contesto determinato;
- un URI o un IRI deve poter essere ricordato facilmente dalle persone, per cui è utile che la stringa che rappresenta un URI abbia un significato che ne faciliti la memorizzazione.

Dal momento che ci deve essere la possibilità di rappresentare un URI o un IRI all'interno di parentesi di qualsiasi tipo, i caratteri corrispondenti a queste parentesi non possono essere utilizzati letteralmente all'interno di un indirizzo del genere. Le parentesi in questione sono quelle tonde, quadre, graffe e angolari: '**(**', '**)**', '**[**', '**]**', '**{**', '**}**', '**<**', '**>**'.

54.1.2 Sintassi

La sintassi di un URI o un IRI è piuttosto complessa, perché dipende molto dal contesto a cui si applica. Non è il caso si entrare troppo nel dettaglio; piuttosto è meglio apprendere la logica della cosa.

schema **:** ***parte_successiva_dipendente_dallo_schema***

Quello che si vede è il modello di prima approssimazione di un indirizzo URI o IRI assoluto (viene trattato in seguito il concetto di URI/IRI relativo). In questa prima fase si distinguono due parti, separate da due punti verticali ('**:**'), dove prima appare un nome che definisce uno «schema» e poi continua con una stringa che va interpretata in base alle regole specifiche di quello schema.

La sintassi di un URI/IRI non stabilisce a priori quale sia la forma che deve avere la stringa che segue i due punti; tuttavia, è frequente l'utilizzo di URI secondo i modelli seguenti:

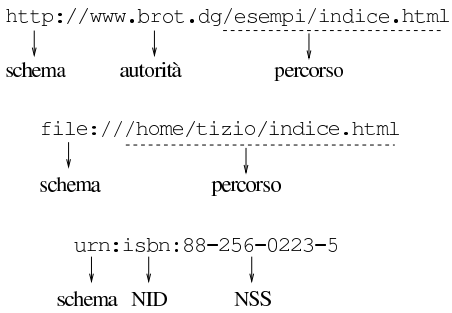
schema **:** ***//autorità*** **[** ***percorso*** **[** **?** ***interrogazione*** **]]**

schema **:** ***/percorso***

Convenzionalmente, quando una risorsa viene individuata attraverso un URI/IRI che per sua natura contiene un'informazione gerarchica, la separazione tra i vari livelli di questa gerarchia avviene utilizzando una barra obliqua normale ('**/**'). Si tratta evidentemente di una tecnica ereditata dal file system Unix; tuttavia, ciò resta indipendente dal fatto che la risorsa in questione risieda fisicamente all'interno di un file system o meno.

La figura 54.1 mostra alcuni esempi a proposito di URI/IRI composti secondo i modelli più frequenti.

Figura 54.1. Esempi di URI/IRI comuni.



Nella figura si vede anche un caso particolare, riferito a un URN di tipo ISBN (*International standard book number*). Lo schema di un URN è sempre '**urn:**'; a questo segue l'indicazione di un NID (*Namespace identifier*), ovvero un identificatore che qualifica l'informazione successiva; infine si inserisce l'informazione, definita NSS (*Namespace specific string*), ovvero ciò che va inteso nel contesto stabilito dal NID. L'esempio che appare nella figura fa riferimento al numero ISBN 88-256-0223-5, esprimendolo in forma di URN.

54.1.2.1 Accesso a un servente attraverso la rete

Quando l'indirizzo URI si riferisce a un servizio offerto attraverso la rete, la struttura di ciò che è stato definito come «autorità» si articola in modo particolare:

[***utente*** **[** **:** ***parola_d'ordine*** **]@]** ***nodo*** **[** **:** ***porta*** **]**

In questo modo si può specificare il nominativo utente per l'accesso alla risorsa, eventualmente anche la parola d'ordine (benché ciò sia decisamente sconsigliabile per motivi di sicurezza), quindi il nodo di rete che offre il servizio e infine la porta del servizio.

Il nodo di rete può essere indicato per nome, attraverso il nome a dominio, oppure attraverso il numero IPv4. Purtroppo non è sta-

to definito un modo per indicare un numero IPv6, dal momento che la sua forma renderebbe impossibile l'interpretazione corretta dell'indirizzo.

Se si omettono le informazioni riferite all'utente, vuol dire che queste non sono necessarie, oppure che esistono dei valori predefiniti per questo; per quanto riguarda la porta del servizio, se questa non viene indicata si fa riferimento sempre al suo valore predefinito. Naturalmente, è stabilito dal servente quali siano i valori predefiniti.

54.1.2.2 Riferimento agli URI

Per sua natura, l'indirizzo URI è un riferimento a una risorsa. In generale vanno considerate anche due circostanze particolari: il riferimento a un frammento della risorsa e l'indicazione di URI relativi.

Un URI relativo è un indirizzo ridotto che parte da un punto di partenza conosciuto. Il principio deriva dal concetto di percorso relativo all'interno di un file system. In generale, un URI relativo può essere indicato omettendo tutta la parte iniziale che si possa determinare altrimenti.

Di fronte a un URI che contenga un'informazione sul percorso in forma gerarchica, è abbastanza facile intendere cosa sia la base di riferimento per gli URI relativi: basta togliere dall'indirizzo attuale tutto quello che segue l'ultima barra obliqua. Per esempio, per il documento *http://www.brot.dg/esempi/articolo.html* l'URI di base è *http://www.brot.dg/esempi/*, per cui, il riferimento a '`figure/foto.jpg`' richiama effettivamente l'URI *http://www.brot.dg/esempi/figure/foto.jpg*.

Il percorso di un URI relativo può essere indicato anche con una barra obliqua iniziale, ma in questo caso si intende fare riferimento a un percorso assoluto nell'ambito dell'URI. Continuando con l'esempio precedente, il riferimento a '`/nuovo/documento.html`' richiama effettivamente l'URI *http://www.brot.dg/nuovo/documento.html*.

In presenza di un percorso relativo, è possibile utilizzare anche i simboli '`.`' e '`..`', con lo stesso significato che hanno nel file system Unix: il primo rappresenta la posizione corrente e il secondo quella precedente.

È importante osservare che il riferimento alla stringa nulla indica implicitamente lo stesso URI iniziale.

Il problema degli URI relativi non è così semplice come è stato descritto. In realtà vanno prese in considerazione altre cose, come per esempio la possibilità che il tipo di risorsa (di solito in un documento HTML) possa incorporare l'informazione esplicita di un URI di base.

Quando il tipo di risorsa lo consente, è possibile aggiungere all'URI l'indicazione di un frammento particolare. Questa parte aggiuntiva la si riconosce perché è preceduta dal simbolo '`#`':

```
http://www.brot.dg/esempi/articolo.html#commento
```

L'esempio mostra il riferimento al frammento '**`#commento`**' nell'ambito dell'URI '`http://www.brot.dg/esempi/articolo.html`'. Dal momento che la stringa nulla fa riferimento alla risorsa attuale, i riferimenti interni alla stessa risorsa sono indicati facilmente attraverso il solo frammento:

```
#commento
```

L'esempio mostra un riferimento relativo al frammento '**`#commento`**' della risorsa corrente.

54.1.2.3 Esempi

Frequentemente, il nome dello schema dell'indirizzo URI corrisponde al nome del protocollo necessario per raggiungere la risorsa relativa. I più comuni sono: '**`http`**', '**`ftp`**', '**`gopher`**', '**`mailto`**', '**`wais`**', '**`telnet`**', '**`tn3270`**', '**`news`**'. Quando si vuole fare riferimento a un file locale senza utilizzare alcun protocollo particolare, si può indicare anche lo schema '**`file`**', ma in questo caso ci sono delle particolarità che vengono mostrate dagli esempi.

- `http://www.brot.dg:8080/esempi/indice.html`
  - protocollo HTTP
  - nodo di rete *www.brot.dg*
  - porta 8080
    Viene indicata la porta perché si vuole fare riferimento a un valore diverso dallo standard che per il protocollo HTTP è 80
  - risorsa '`/esempi/indice.html`'
- `http://www.brot.dg/esempi/indice.html`
  Come nell'esempio precedente, ma senza l'indicazione della porta che questa volta corrisponde al valore predefinito, cioè 80.
- `http://192.168.1.1/esempi/indice.html`
  Come nell'esempio precedente, ma l'indicazione del nodo avviene per mezzo del suo indirizzo IPv4 invece che attraverso il nome a dominio.
- `ftp://ftp.brot.dg/pub/archivi/esempio.tar.gz`
  - protocollo FTP
  - nodo di rete *ftp.brot.dg*
  - risorsa '`/pub/archivi/esempio.tar.gz`'
- `ftp://tizio@ftp.brot.dg/pub/archivi/esempio.tar.gz`
  Come nell'esempio precedente, con la differenza che si fa riferimento a un utente particolare.
- `ftp://tizio:segretissima@ftp.brot.dg/pub/archivi/esempio.tar.gz`
  Come nell'esempio precedente, con la differenza che si aggiunge l'indicazione della parola d'ordine di accesso al servizio, cosa che in generale è bene non passare mai in questo modo.
- `file://localhost/home/daniele/indice.html`
  In questo caso si vuole fare riferimento a un file locale. Precisamente si tratta del file '`/home/daniele/indice.html`' contenuto nell'elaboratore *localhost*.
  Questo tipo di indicazione è utile specialmente quando si vuole fare riferimento a una pagina indice o iniziale, caricata automaticamente all'atto dell'avvio di un programma cliente per la navigazione.
- `file:///home/daniele/indice.html`
  Esattamente come nell'esempio precedente, con la differenza che si omette l'indicazione esplicita dell'elaboratore locale: *localhost*.
- `file:/home/daniele/indice.html`
  Esattamente come nell'esempio precedente, con la differenza che si utilizza una sola barra obliqua dopo l'indicazione '**`file:`**' (ma in generale è preferibile la forma precedente, con le tre barre oblique).
- `mailto:tizio@dinkel.brot.dg`
  Si tratta di un indirizzo di posta elettronica, nel quale è essenziale fornire l'indicazione del nominativo utente. Dopo il nome del nodo di destinazione non appare un percorso, perché in questo caso non avrebbe significato.

54.1.3 Limitazioni nell'uso dei caratteri

Ogni componente di un URI ha delle regole proprie nell'uso dei caratteri, dal momento che alcuni di questi hanno significati speciali. Purtroppo le regole in questione sono tante e la cosa migliore che si può fare è quella di usare il buon senso, riservando la lettura della documentazione specifica ai casi in cui è indispensabile chiarire il problema nel dettaglio (RFC 2396).

In generale non è ammissibile l'uso dello spazio. Infatti, considerato il principio di trascrivibilità degli URI, lo spazio dovrebbe essere inteso solo come una necessità legata al tipo di trascrizione utilizzata. Per il resto, se la propria lingua lo consente, sarebbe bene limitarsi all'uso delle lettere dell'alfabeto latino (maiuscole e minuscole, ma senza accenti), le cifre numeriche e alcuni simboli: '**@**', '*****', '**_**', '**-**' e il punto ('**.**'). Gli altri simboli possono creare problemi di trascrivibilità o avere significati particolari (basta pensare alle barre oblique e ai due punti verticali).

Quando un simbolo particolare non può essere utilizzato in modo letterale nel contesto in cui lo si vuole inserire, può essere indicato attraverso una notazione speciale: '***%hh***'. La sigla ***hh*** rappresenta una coppia di cifre esadecimali. A questa regola fa eccezione lo spazio che viene codificato normalmente con il segno '**+**', ma non in tutte le occasioni (di solito solo nelle stringhe di richiesta).

Generalmente, per gli indirizzi URI normali non c'è la necessità di preoccuparsi di questo problema, anche la tilde può essere utilizzata letteralmente nell'indicazione dei percorsi. La tabella 54.4 mostra l'elenco di alcune corrispondenze tra simboli particolari e la codifica alternativa utilizzabile negli URI.

Tabella 54.4. Alcune corrispondenze tra simboli particolari e codifica alternativa utilizzabile negli URI.

Carattere	Codifica corrispondente
%	%25
&	%26
+	%2B
/	%2F
=	%3D

In linea di principio, un URI dovrebbe essere realizzato in modo da non dover utilizzare questa tecnica di protezione per i caratteri «speciali». La situazione più probabile in cui è necessario utilizzare questo procedimento è riferito alle stringhe di interrogazione.

54.2 Aspetti generali di HTML

HTML sta per *Hypertext markup language* e in pratica è un formato SGML per i documenti della rete che fa uso di un DTD particolare: HTML appunto. La composizione di un documento HTML non può mai essere valutata perfettamente in anticipo, perché dipende da diversi fattori:

- il programma utilizzato per visualizzare il documento;
- la risoluzione utilizzata;
- i tipi di carattere a disposizione;
- la profondità di colori disponibili.

Lo standard HTML è tale per cui tutti (o quasi) i programmi utilizzabili per la lettura di tali documenti sono in grado di cavarsela. Ma questo risultato minimo è ben lontano dall'esigenza di costruire qualcosa che tutti possano vedere più o meno nello stesso modo. Per questo, quando si costruisce un documento HTML, occorre pensare all'utenza a cui è destinato, in modo da decidere quali caratteristiche possono essere utilizzate e quali invece è meglio scartare per evitare inutili problemi di lettura.

L'HTML nasce all'inizio degli anni 1990, abbinato in particolare al primo navigatore: Mosaic. Da quel momento a oggi il formato

HTML ha subito diversi aggiornamenti; si ricorda in particolare la versione 2.0 del 1995 e la versione 3.2 del 1997. Secondo il progetto di W3C, lo sviluppo di HTML avrebbe dovuto concludersi con la versione 4.01, per passare da quel momento a XHTML. Tuttavia, intorno al 2007 la questione di HTML viene riaperta, introducendo una versione 5. Tra le varie versioni di HTML va tenuta in considerazione particolare quella definita dallo standard ISO 15445, la quale può considerarsi un passepartout per ogni tipo di navigatore ipertestuale.

Lo spirito alla base dello sviluppo dell'HTML da parte del W3C, così come acquisito dallo standard ISO 15445, è quello di ottenere un formato multimediale-ipertestuale completo, adatto per la lettura attraverso qualunque tipo di mezzo: dal terminale tattile braille al documento stampato. Le estensioni proprietarie di questo standard si sono rivolte principalmente all'aspetto visuale e scenografico del formato, trascurando le altre esigenze. Scrivere un documento «puro» in HTML è un'arte raffinata, ma poco conosciuta. In generale, **maggiori sono i contenuti e le esigenze di divulgazione, minori devono essere le pretese estetiche**.

La documentazione di riferimento per tutto ciò che riguarda l'HTML è quella offerta dal W3C: *http://www.w3.org* , in particolare *http://www.w3.org/TR/* , a cui si deve affiancare anche quanto riguarda lo standard ISO 15445: *http://www.scss.tcd.ie/misc/15445/15445.html* .

54.2.1 HTML e SGML

L'HTML è un linguaggio di composizione basato sull'SGML (si veda quanto descritto nel capitolo 51). Come tale, un documento HTML inizia sempre con la dichiarazione del DTD; poi tutto il documento viene racchiuso nell'elemento principale di questa struttura:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
    ...
    ...
    ...
</HTML>
```

Purtroppo, la maggior parte dei programmi di navigazione o di composizione per il formato HTML non è in grado di comprendere tutte le regole dell'SGML, per cui occorre evitare di utilizzare alcune delle sue caratteristiche. In particolare bisogna evitare:

- la creazione di entità generali per l'utilizzo di macro specifiche relative al testo;

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN"
[
<!ENTITY GNULINUX "GNU/Linux">
<!ENTITY HURD     "GNU/Hurd">
<!ENTITY DOS      "Dos">
<!ENTITY POSIX    "POSIX">
<!ENTITY UNIX     "Unix">
]>
```

- le sezioni marcate per l'inclusione e l'esclusione del testo;

```
<![INCLUDE[
    …
    <!-- testo incluso -->
    …
]]>
…
<![IGNORE[
    …
    <!-- testo escluso -->
    …
]]>
```

- le sezioni marcate per individuare un contenuto di tipo '**CDATA**', allo scopo di proteggere il simbolo '**<**' in modo da poterlo usare letteralmente;

```
<![CDATA[
    …
    <!-- testo letterale -->
    …
]]>
```

• la delimitazione di un elemento in forma abbreviata;

```
<nome_elemento / contenuto_dell'elemento /
```

• l'indicazione di marcatori iniziali e finali vuoti.

```
<> ... </>
```

Il fatto che l'HTML sia definito da un DTD, permette di verificare la sua correttezza formale, anche se le regole stabilite nel DTD non sono sufficienti a definire la sintassi completa. Per poter verificare la correttezza formale di un documento HTML, oltre agli strumenti di convalida, cioè il pacchetto SP, occorre procurarsi il DTD e le sue estensioni riferite alle entità generali, quelle che permettono di utilizzare le macro per le lettere accentate e i simboli speciali.

Il DTD dell'HTML ISO 15445 e la dichiarazione SGML si trovano presso *https://www.scss.tcd.ie/misc/15445/15445.HTML* (da cui vanno estrapolati i file '`15445.dcl`' e '`15445.dtd`'). Per quanto riguarda le entità standard a cui si fa riferimento, queste si trovano presso *http://www.w3.org/TR/html4/* . Si può realizzare un catalogo SGML per l'analisi locale di un documento del genere nel modo seguente:

```
OVERRIDE YES

SGMLDECL                                                    15445.dcl

PUBLIC  "ISO/IEC 15445:2000//DTD HyperText Markup Language//EN" 15445.dtd
DTDDECL "ISO/IEC 15445:2000//DTD HyperText Markup Language//EN" 15445.dcl
PUBLIC  "ISO/IEC 15445:2000//DTD HTML//EN"                  15445.dtd
DTDDECL "ISO/IEC 15445:2000//DTD HTML//EN"                  15445.dcl
PUBLIC "-//W3C//ENTITIES Full Latin 1//EN//HTML"            HTMLlat1.ent
PUBLIC "-//W3C//ENTITIES Special//EN//HTML"                 HTMLspecial.ent
PUBLIC "-//W3C//ENTITIES Symbolic//EN//HTML"                HTMLsymbol.ent
```

Con questo catalogo, i file utilizzati devono trovarsi nella directory corrente.

Nell'esempio seguente si utilizza il programma '**`nsgmls`**' (del pacchetto SP) supponendo in particolare che il catalogo sia contenuto nel file '`catalogo`'; il file da verificare viene indicato come '`mio_file.html`'. Il catalogo e il file da controllare si intendono collocati nella directory corrente.

```
$ cat mio_file.html | nsgmls -s -c catalogo [Invio]
```

È il caso di ricordare che alcune distribuzioni GNU/Linux, in particolare Debian, predispongono un pacchetto apposito contenente i DTD più comuni riferiti alle varie versioni dell'HTML, comprese le estensioni proprietarie, assieme alle relative entità standard. Naturalmente, il tutto è organizzato in un catalogo unico che va eventualmente ad aggiornare il catalogo di sistema (dovrebbe trattarsi del file '`/etc/sgml.catalog`', oppure del file '`/usr/share/sgml/catalog`'). Il nome di questo pacchetto potrebbe essere '`sgml-data*`'.

Oltre alla verifica in base al DTD sarebbe opportuno sapere leggere il contenuto del DTD stesso. A questo proposito è da notare il fatto che nel manuale che descrive le specifiche HTML ISO 15445, si fa spesso riferimento alle caratteristiche degli elementi attraverso lo schema offerto dalla dichiarazione relativa nel DTD. In effetti, ciò permette di rendere molto chiara e precisa la descrizione che ne viene fatta subito dopo.

54.2.1.1 Attributi comuni attraverso le entità parametriche

Il DTD dell'HTML ISO 15445 fa un uso massiccio di entità parametriche e questo può disorientare inizialmente. In generale basta ricordare che qualunque cosa nella forma '***%nome;***' è una macro che si espande in una stringa. La dichiarazione di queste entità parametriche avviene nella parte iniziale del DTD, attraverso istruzioni del tipo:

```
<!ENTITY % nome "stringa">
```

È interessante notare l'utilizzo di entità parametriche per fare riferimento agli attributi degli elementi. Infatti, quasi tutti gli elementi

dell'HTML ISO 15445 prevedono l'uso di attributi, per cui si è ritenuto opportuno classificarli all'interno di entità parametriche. In particolare è importante individuarne due molto importanti:

```
<!ENTITY % core
   "CLASS      CDATA      #IMPLIED -- Comma separated list of class values --
    --The name space of the ID attribute is shared with the name space of
      the NAME attributes.  Both ID and NAME attributes may be provided for
      the <A> and <MAP> elements. When both ID and NAME values are provided
      for an element, the values shall be identical.  It is an error for an
      ID or NAME value to be duplicated within a document.

      It is recommended that authors of documents specify both the ID
      attribute and the NAME attribute for the <A> and <MAP> elements.
    --
    ID         ID         #IMPLIED -- Document-wide unique id --
    TITLE      CDATA      #IMPLIED -- Advisory title or amplification --" >

        <!-- Internationalization attributes -->

<!ENTITY % i18n
   "DIR        (ltr|rtl)  #IMPLIED -- Direction for weak/neutral text --
    LANG       NAME       #IMPLIED -- RFC1766 language value --" >
```

La macro '**`%core;`**' serve a individuare un gruppo di attributi disponibili nella maggior parte degli elementi:

Attributo	Descrizione
`ID`	permette di attribuire una stringa di riconoscimento all'elemento, in modo da potervi fare riferimento;
`CLASS`	permette di abbinare all'elemento una classe, definita attraverso un nome, in modo da potergli attribuire uno stile particolare;
`TITLE`	permette di attribuire un «titolo» all'elemento, cosa che si traduce in pratica in modo differente a seconda del contesto (ovvero, a seconda dell'elemento a cui si applica).

La macro '**`%i18n;`**' serve invece a definire ciò che riguarda la localizzazione:

Attributo	Descrizione
`LANG`	permette di indicare una sigla, secondo lo standard ISO 639 (tabella 13.4) e anche secondo altri standard, per attribuire all'elemento il linguaggio relativo;
`DIR`	permette di stabilire il flusso del testo nel risultato finale, dove la parola chiave '**`ltr`**' si riferisce a uno scorrimento da sinistra a destra (*Left to right*) e la parola chiave '**`rtl`**' indica uno scorrimento opposto, da destra a sinistra (*Right to left*).

Si osservi, a titolo di esempio, la dichiarazione dell'elemento '**`P`**', dove gli attributi sono quelli più comuni, rappresentati dalle macro '**`%core;`**' e '**`%i18n;`**':

```
<!ELEMENT P     - O  (%text;)+ >
<!ATTLIST P
    %core;          -- Element CLASS, ID and TITLE --
    %i18n;          -- Internationalization DIR and LANG -->
```

54.2.1.2 Classificazione fondamentale degli elementi

All'interno di un documento HTML si distinguono due gruppi di elementi fondamentali: quelli che rappresentano dei blocchi e quelli che servono a inserire qualcosa all'interno di una riga di testo normale. Questa suddivisione corrisponde a due macro: '**`%block;`**' e '**`%text;`**' rispettivamente.

Per fare un esempio, l'elemento '**`P`**' (paragrafo) è un «blocco», mentre l'elemento '**`EM`**' (enfasi) è un componente interno a una riga di testo.

Questa classificazione semplifica molto la dichiarazione degli elementi, come nel caso dell'elemento '**`P`**', già visto, il cui contenuto è semplicemente tutto ciò che va inserito nelle righe di testo:

```
<!ELEMENT P     - O  (%text;)+ >
```

Alcuni elementi di un documento HTML sono ambigui, nel senso che possono contenere sia blocchi che testo. A titolo di esempio si

osservi la dichiarazione dell'elemento '**LI**' che rappresenta la voce di un elenco puntato o numerato:

```
<!ELEMENT LI      - O  (%text; | %block;)+ >
```

54.2.2 Stili

Le estensioni proprietarie dell'HTML hanno portato questo linguaggio di composizione a una proliferazione di dialetti, a causa dell'esigenza di trasferire anche le informazioni sull'aspetto finale della composizione, che in origine non sono state prese in considerazione. L'unica soluzione disponibile con HTML ISO 15445 è l'abbinamento di uno stile, che può essere dichiarato all'interno del file HTML stesso, attraverso l'elemento '**STYLE**', oppure in un file esterno, richiamandolo con l'elemento '**LINK**' (viene mostrato tra poco).

L'HTML non presuppone il formato in cui può essere realizzato lo stile. È comune l'uso di stili in formato CSS (*Cascading style sheet*) e per farvi riferimento si indica il tipo '**text/css**'.

Per il momento, non viene spiegato in che modo si scrivono le direttive in un foglio di stile CSS. Intuitivamente, il lettore può comprendere che la direttiva seguente serve a colorare in blu il contenuto degli elementi '**H1**':

```
H1 { color: blue; }
```

Inoltre, la direttiva seguente serve per fare in modo che il contenuto dell'elemento '**P**' abbia il carattere di 12 punti e di colore rosso:

```
P { font-size: 12pt; color: red; }
```

Si osservi che la stessa cosa avrebbe potuto essere scritta nel modo seguente:

```
P {
    font-size:  12pt;
    color:      red;
}
```

Per definire questi stili all'interno di un documento HTML, senza fare uso di un file esterno, si potrebbe agire nel modo seguente, attraverso l'uso dell'elemento '**STYLE**':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <STYLE TYPE="text/css">
        H1 { color: blue }
        P {
            font-size:  12pt;
            color:      red;
        }
    </STYLE>
</HEAD>
<BODY>
    ...
    ...
    ...
</BODY>
</HTML>
```

Si comprende che il testo contenuto nell'elemento '**STYLE**' non deve interferire con l'HTML e quindi non può contenere simboli che possano risultare ambigui. Questo problema riguarda naturalmente il linguaggio con cui è realizzato lo stile; nel caso del formato CSS non dovrebbe porsi alcun problema. Tuttavia, qualche programma utilizzato per la navigazione, potrebbe non riconoscere l'elemento '**STYLE**', arrivando a riprodurre il testo che rappresenta in realtà lo stile. Per evitare questo problema si può circoscrivere la cosa all'interno di un commento SGML:


```
<STYLE TYPE="text/css">
    <!--
        H1 { color: blue }
        P {
            font-size:  12pt;
            color:      red;
        }
    -->
</STYLE>
```

Probabilmente, il modo più elegante di abbinare uno stile a un documento HTML è quello di aggiungere un file esterno. Nell'esempio seguente si include lo stile corrispondente al file '`stile.css`':

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Esempio</TITLE>
    <LINK REL="stylesheet" TYPE="text/css" HREF="stile.css">
    ...
</HEAD>
...
</HTML>
```

È chiaro che dipende dal programma di navigazione la capacità o meno di conformarsi allo stile. In generale, lo standard CSS sembra essere quello che ha più probabilità di affermarsi.

54.2.3 Struttura di un documento HTML

Il documento HTML è contenuto tutto nell'elemento omonimo: '**HTML**'. Questo si scompone in due elementi fondamentali, '**HEAD**' e '**BODY**', che rappresentano rispettivamente l'intestazione e il corpo:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML>
<HEAD>
    <TITLE>Titolo della pagina</TITLE>
</HEAD>
<BODY>
    ...
    <!-- Corpo del documento -->
    ...
</BODY>
</HTML>
```

In generale, è conveniente annotare la lingua principale del documento, attraverso l'attributo '**LANG**' da collocare nel marcatore di apertura dell'elemento '**HTML**':

```
<HTML LANG="it">
```

Per la precisione, il codice che definisce il linguaggio viene indicato secondo la sintassi seguente:

codice_principale [- ***codice_secondario***]

In pratica, la prima parte, quella che appare prima del trattino di separazione, indica la lingua, di solito attraverso il codice ISO 639 (tabella 13.4), mentre la seconda parte indica l'area nazionale, secondo lo standard ISO 3166 (tabella 13.5), che a sua volta può implicare delle varianti nel linguaggio.

In generale, un documento di grandi dimensioni realizzato attraverso il formato HTML, richiede la scomposizione dello stesso in più file HTML collegati tra loro da riferimenti ipertestuali. Questa, purtroppo, è una necessità a causa delle limitazioni dei programmi di navigazione.

54.2.3.1 Intestazione e informazioni supplementari

L'intestazione è una parte del documento HTML che serve per annotare delle informazioni generali. Deve contenere almeno il titolo all'interno dell'elemento '**TITLE**'. Di solito, la riproduzione di un documento HTML non fa apparire il titolo nel testo del documento, che comunque viene usato per farvi riferimento (per esempio nel segnalibro del programma utilizzato per la sua visualizzazione).

Nell'intestazione, prima o dopo il titolo, può essere conveniente collocare alcune «meta-informazioni», attraverso alcuni elementi '**META**'. Si tratta di un elemento vuoto, per il quale si utilizza soltanto il marcatore di apertura con l'indicazione di attributi opportuni. In particolare, si possono utilizzare gli attributi seguenti:

Attributo	Descrizione
NAME	per indicare un nome che qualifica il tipo di meta-informazione (si tratta di parole chiave più o meno standard, che però non sono state definite nel DTD);
HTTP-EQUIV	per indicare un campo di risposta nell'ambito del protocollo HTTP, tenendo conto che l'attributo '**NAME**' è alternativo a '**HTTP-EQUIV**';
CONTENT	(obbligatorio) per indicare il valore abbinato al nome indicato attraverso l'attributo '**NAME**', oppure attraverso l'attributo '**HTTP-EQUIV**'.

Come si intuisce dall'elenco degli attributi più importanti, si può distinguere tra elementi '**META**' che utilizzano l'attributo '**NAME**' e altri che usano l'attributo '**HTTP-EQUIV**'. Le informazioni che si definiscono attraverso elementi '**META**' con l'attributo '**NAME**' permettono di indicare informazioni che qualificano il documento, soprattutto quando questo viene trattato automaticamente da un motore di ricerca; l'attributo '**HTTP-EQUIV**' permette invece di intervenire a livello del protocollo HTTP (quando il documento viene ottenuto in questo modo), specificando le intestazioni HTTP relative. Si osservi l'esempio seguente:

```
<HEAD>
    <TITLE>Titolo della pagina</TITLE>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=UTF-8">
    <META NAME="Description"
          CONTENT="Esempio di una pagina HTML">
    <META NAME="Keywords"
          CONTENT="HTML, SGML, Editoria elettronica">
    <META NAME="Author"
          CONTENT="P. Pallino ppallino@dinkel.brot.dg">
    <META NAME="Classification" CONTENT="Esempio HTML">
</HEAD>
```

In particolare, ricevendo questo documento attraverso il protocollo HTTP, si ottiene anche l'intestazione HTTP seguente:

```
Content-Type: text/html; charset=UTF-8
```

Si noti nell'esempio l'indicazione esplicita dell'insieme di caratteri: UTF-8.

Un altro tipo di elemento speciale può apparire all'interno dell'intestazione di un documento HTML; si tratta di '**LINK**'. Anche questo è un elemento vuoto e serve solo per indicare degli attributi nel marcatore di apertura. Gli attributi più importanti sono:

Attributo	Descrizione
HREF	per indicare un URI a cui si intende fare riferimento;
REL	per definire la relazione che c'è con questo tipo di collegamento;
TYPE	per specificare in anticipo il tipo dei dati contenuti nell'URI;
MEDIA	per specificare il mezzo attraverso cui viene letto il documento.

Trattandosi di un elemento vuoto, collocato nell'intestazione HTML, non è pensato per essere rappresentato nella composizione. Tuttavia, abbinando le parole chiave opportune all'attributo '**REL**', si stabiliscono dei collegamenti utili per ricomporre un documento più grande costituito da più pagine HTML. In pratica, si può dichiarare in modo esplicito come è articolato, così che il programma di navigazione o composizione sappia regolarsi. La tabella 54.29 elenca alcune delle parole chiave che possono essere assegnate all'attributo '**REL**'.

Tabella 54.29. Parole chiave tipiche da assegnare all'attributo '**REL**' dell'elemento '**LINK**'.

Nome	Descrizione
Alternate	Una versione alternativa dello stesso documento.
Stylesheet	Foglio di stile esterno.
Start	Il primo documento di una collezione.
Next	Il prossimo documento di una sequenza lineare.
Prev	Il documento precedente di una sequenza lineare.
Contents	Un documento che funge da indice generale.
Index	Un documento che funge da indice analitico.
Glossary	Un documento che funge da glossario.
Copyright	Un documento che contiene la dichiarazione del copyright.
Chapter	Un documento che funge da capitolo in una collezione.
Section	Un documento che funge da sezione in una collezione.
Subsection	Un documento che funge da sottosezione in una collezione.
Appendix	Un documento che funge da appendice in una collezione.
Help	Un documento che funge da guida.

L'esempio seguente mostra parte di un'intestazione di una pagina HTML in cui sono stati usati alcuni elementi '**LINK**' per definire la relazione con altre pagine che compongono la stessa raccolta:

```
<HEAD>
    ...
    <LINK REL="Stylesheet" TYPE="text/css" HREF="stile.css">
    <LINK REL="Start" TITLE="inizio" HREF="index.html">
    <LINK REL="Contents" TITLE="indice generale"
          HREF="indice-generale.html">
    <LINK REL="Prev" TITLE="precedente"
          HREF="capitolo-6.html">
    <LINK REL="Next" TITLE="successivo"
          HREF="capitolo-8.html">
    <LINK REL="Index" TITLE="indice analitico"
          HREF="indice-analitico.html">
</HEAD>
```

Merita un po' di attenzione l'attributo '**MEDIA**' che serve a stabilire il mezzo adatto per la lettura del documento relativo. Questo attributo si usa generalmente all'interno di un elemento '**LINK**' che serve a indicare un foglio di stile esterno; inoltre può essere usato per lo stesso motivo all'interno di un elemento '**STYLE**'. In pratica, in questo modo, si stabilisce l'abbinamento tra stile e mezzo di lettura. La tabella 54.31 elenca i nomi che si possono assegnare a un attributo '**MEDIA**'.

Tabella 54.31. Parole chiave tipiche da assegnare all'attributo '**MEDIA**' dell'elemento '**LINK**' e dell'elemento '**STYLE**'.

Nome	Descrizione
screen	Schermo per lo scorrimento continuo.
tty	Terminale a celle di caratteri o simile.
tv	Televisione (bassa risoluzione e altre limitazioni).
projection	Proiettore.
handheld	Schermi portatili.
print	Stampa e simili (composizione impaginata).
braille	Terminale a barra braille per i non vedenti.
aural	Lettore a sintesi vocale.
all	Valido per tutti i tipi di dispositivo.

L'esempio seguente mostra in che modo si potrebbero selezionare diversi fogli di stile in base al mezzo utilizzato per la lettura del documento:

```
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="screen"
      HREF="stile-schermo.css">
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="tty"
      HREF="stile-testo.css">
<LINK REL="Stylesheet" TYPE="text/css" MEDIA="braille"
      HREF="stile-braille.css">
```

54.2.3.2 Corpo del documento

Il corpo di un documento HTML è delimitato dall'elemento '**BODY**' e il suo contenuto è ciò che alla fine viene mostrato nella composizione finale. La composizione del corpo viene descritta nella sezione 54.4.

54.3 Attributi comuni

Si è già accennato al fatto che molti elementi condividano un insieme comune di attributi. Vale la pena di descrivere brevemente alcuni di questi nelle sezioni successive.

54.3.1 Linguaggio

Il linguaggio di un elemento viene definito esplicitamente attraverso l'attributo '**LANG**', a cui viene assegnato solitamente un codice corrispondente allo standard ISO 639. La tabella 54.33 riporta un elenco di questi codici ridotto ad alcune lingue occidentali.

Tabella 54.33. Alcuni codici dello standard ISO 639 per la definizione della lingua attraverso una sigla di due soli caratteri.

Codice	Lingua	Codice	Lingua
fr	Francese	it	Italiano
ro	Rumeno	es	Spagnolo
ca	Catalano	co	Corso
pt	Portoghese	da	Danese
nl	Olandese	en	Inglese
de	Tedesco	is	Islandese
no	Norvegese	sv	Svedese
fi	Finlandese		

In generale può essere conveniente l'utilizzo di questo attributo nell'elemento '**HTML**', in modo da fissare il linguaggio di tutto il documento. Tuttavia, quando un elemento contiene un testo in un altro linguaggio, conviene annotarlo nello stesso modo.

L'effetto più evidente che potrebbe risultare dalla distinzione in base al linguaggio, è la separazione delle parole in sillabe, per creare una composizione più gradevole.

54.3.2 Codifica

L'opzione '**charset**' dell'attributo '**CONTENT**', permette di definire esplicitamente l'insieme di caratteri dell'elemento. Come è già stato mostrato, di solito lo si utilizza in un elemento '**META**' introduttivo allo scopo di definire l'intestazione HTTP relativa:

```
<HEAD>
    <TITLE>...</TITLE>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=UTF-8">
    <!--...-->
</HEAD>
```

La tabella 54.35 elenca alcuni codici comuni per la definizione dell'insieme dei caratteri.

Tabella 54.35. Alcuni codici per definire l'insieme di caratteri.

Codice	Corrispondenza	Codice	Corrispondenza
ISO−8859−1	latin1	ISO−8859−2	latin2
ISO−8859−3	latin3	ISO−8859−4	latin4
ISO−8859−5	ciryllic	ISO−8859−6	arabic
ISO−8859−7	greek	ISO−8859−8	hebrew
ISO−8859−9	latin5	ISO−8859−15	latin9
UTF-8	codifica universale compatibile con ASCII.	UTF-16	

54.3.3 Direzione del testo

Il testo di un documento HTML può scorrere da sinistra a destra o viceversa. Per controllare questo flusso si può utilizzare l'attributo '**DIR**', a cui si possono abbinare esclusivamente le parole chiave '**LTR**' o '**RTL**': *Left to right*, da sinistra a destra; *Right to left*, da destra a sinistra.

In generale, il flusso del testo avviene da sinistra a destra, come richiedono le lingue occidentali, per cui non è necessario usare questo attributo in condizioni «normali».

È importante notare che il testo nel sorgente di un documento HTML segue sempre il flusso normale, da sinistra a destra, ammesso che si possa definire un flusso per un file sorgente.

Non è disponibile la possibilità di ribaltare orizzontalmente i caratteri, quando il flusso del testo cambia direzione, come avviene nella scrittura geroglifica.

54.3.4 Titolo

Molti elementi dispongono di un attributo '**TITLE**'. Il suo scopo è quello di indicare un titolo, che viene preso in considerazione in modo differente in base al contesto. Questo attributo può essere molto utile negli elementi che comportano l'inclusione di un'immagine, dal momento che rappresenta un testo alternativo per chi non può visualizzarle. Anche un riferimento ipertestuale può avvantaggiarsi di questo attributo, perché si può visualizzare il testo corrispondente prima di raggiungere l'oggetto, in modo da avere una breve descrizione di ciò che si tratta (così da poter decidere se ne vale la pena).

```
<A HREF="http://www.brot.dg/foto/tizio.jpg"
   TITLE="Tizio in divisa">Tizio</A>
```

L'esempio mostra proprio il caso di un riferimento ipertestuale, ottenuto con l'elemento '**A**', attraverso il quale si raggiunge un file che dovrebbe mostrare l'immagine di Tizio vestito in divisa. Se il navigatore permette di conoscere il titolo del riferimento prima di doverlo raggiungere, si può evitare di prelevare il file nel caso ciò non sia interessante.

È ovvio che sta poi all'autore della pagina la scelta nello scrivere dei titoli utili o ingannevoli. Chi realizza una pagina pubblicitaria ha ovviamente degli interessi diversi da chi invece vuole realizzare un documento ordinato e facile da consultare.

54.3.5 Identificazione di un elemento

Molti elementi dispongono di un attributo '**ID**' che permette di attribuire loro un'etichetta con la quale poi farvi riferimento. Il modo tradizionale per realizzare dei riferimenti incrociati in HTML è l'uso dell'elemento '**A**', prima con l'attributo '**NAME**' (l'etichetta), poi con l'attributo '**HREF**' (il riferimento ipertestuale).

L'attributo '**ID**' permette di generalizzare il problema, dal momento che in tal modo gli elementi comuni hanno la possibilità di «identificarsi» in maniera univoca per qualunque scopo, non solo quello di definire un obiettivo per un riferimento.

```
<P ID="superparagrafo">Questo è un paragrafo nominato in
modo univoco.</P>
<P ID="supermegaparagrafo">Anche questo è un altro
paragrafo nominato in modo univoco.</P>
```

Si deve tenere presente che i nomi utilizzati per gli attributi '**ID**' devono essere univoci. Questi nomi devono essere univoci anche nei confronti dell'attributo '**NAME**' nell'elemento '**A**'.

54.3.6 Classificazione degli elementi

A differenza dell'attributo '**ID**', l'attributo '**CLASS**' consente di abbinare a un gruppo di elementi una certa classe. Il meccanismo è lo stesso, con la differenza che si vogliono indicare dei raggruppamenti. Di solito, si attribuisce una classe per abbinarne le definizioni di un foglio di stile.

```
<SPAN CLASS="nota">la vita è fatta per essere vissuta</SPAN>
```

L'esempio mostra la delimitazione di una parte di testo attraverso l'elemento '**SPAN**', al quale viene attribuita la classe '**nota**'. In seguito è possibile abbinare a tutti gli elementi di questa classe le stesse caratteristiche attraverso un foglio di stile. Utilizzando i fogli di stile CSS, si potrebbe applicare la regola seguente a tutti gli elementi '**SPAN**' della classe '**nota**':

```
SPAN.nota       { color: green; }
```

54.4 HTML: corpo

Il corpo di un documento HTML è contenuto nell'elemento '**BODY**', che può contenere blocchi di testo, intercalati da elementi '**H*n***' (da '**H1**' a '**H6**'), che rappresentano il titolo di una sezione di livello ***n***. In particolare, lo standard ISO 15445 impone che il livelli delle sezioni siano coerenti.

54.4.1 Delimitazione di blocchi e di testo normale

Per ovviare alla mancanza di una struttura prestabilita, è possibile raggruppare dei blocchi di testo o del testo normale attraverso gli elementi '**DIV**' e '**SPAN**' rispettivamente.

Gli obiettivi che ci si possono prefiggere in questo modo possono essere molti. In generale si sfrutta la possibilità di attribuire a questi elementi degli attributi per qualche scopo.

```
<DIV ID="capitolo-1" CLASS="capitolo">
<!-- contenuto del capitolo -->
...
...
</DIV>
```

Questo esempio mostra una situazione in cui l'elemento '**DIV**' viene utilizzato per delimitare una parte del corpo del documento, a cui viene attribuita la classe '**capitolo**' e la stringa di identificazione '**capitolo-1**'.

```
Il sig. <SPAN CLASS="nome">Tizio Tizi</SPAN> è andato...
```

In questo nuovo esempio, si usa l'elemento '**SPAN**' per delimitare il testo che indica il nome di una certa persona. In questo modo viene anche attribuita l'appartenenza alla classe '**nome**', cosa che può tornare utile per rendere i nomi in modo diverso attraverso un foglio di stile.

54.4.2 Titoli e struttura implicita del testo

Ciò che nel testo rappresenta un titolo di una sezione, si indica utilizzando gli elementi che vanno da '**H1**' a '**H6**'. Intuitivamente, il primo rappresenta un titolo di importanza maggiore, mentre l'ultimo è quello di importanza minore.

L'utilizzo corretto dei titoli attraverso questi elementi è molto importante perché può permettere a un sistema di visualizzazione o composizione di conoscerne la gerarchia e generare così un indice generale (se richiesto). In taluni casi si può arrivare anche a ottenere una numerazione di questi titoli in modo automatico.

```
...
<H1>Titolo principale</H1>
...
<H2>Titolo di livello inferiore</H2>
...
<H1>Altro titolo principale</H1>
...
```

Gli elementi che rappresentano dei titoli sono fatti per contenere testo normale ed elementi che non rappresentano blocchi separati.

È importante ricordare che il titolo del documento HTML, quello che si indica nell'elemento '**TITLE**' nell'intestazione, ovvero all'interno dell'elemento '**HEAD**', di norma non viene rappresentato. Per questo, spesso, il titolo del documento viene riproposto in un titolo '**H1**'.

L'esempio seguente mostra un pezzo di documento strutturato in capitoli e sezioni, delimitati formalmente attraverso l'elemento '**DIV**':

```
<H1>Trattato di bla bla bla</H1>
<DIV CLASS="capitolo" ID="capitolo-1">

<P>Questo trattato tratta di aria fritta...</P>

</DIV>
<H1>Dettagli</H1>
<DIV CLASS="sezione" ID="sezione-1-1">

<P>Questa è una sezione inutile di un trattato
inutile...</P>

</DIV>
```

Lo scopo di ciò può essere quello di attribuire stili particolari alle varie parti gerarchiche del documento. Inoltre, l'utilizzo dell'attributo '**ID**' nell'elemento '**DIV**' che introduce ogni blocco gerarchico può rappresentare un modo coerente per farvi riferimento.

È bene osservare che lo standard ISO 15445 esclude che l'elemento '**DIV**' possa contenere elementi '**H*n***'.

54.4.3 Testo

Il testo normale è ciò che è contenuto in un «blocco» di testo. Il caso più comune di blocco di testo è rappresentato dall'elemento '**P**', utilizzato per dividerlo idealmente in paragrafi.

All'interno di un blocco, salvo casi particolari, il testo viene reso in modo da adattarsi alle dimensioni imposte dal sistema di visualizzazione o di composizione. In pratica, viene suddiviso in modo conveniente, ignorando le interruzioni di riga e le righe vuote aggiunte.

È opportuno fare attenzione all'uso degli spazi all'interno degli elementi che contengono testo normale: si dovrebbe evitare di iniziare o concludere un elemento con uno spazio. In pratica, si deve evitare di scrivere qualcosa come:

```
<P>Bla bla bla <EM> evviva </EM> bla bla.</P>
```

Al suo posto bisogna invece limitarsi a scrivere:

```
<P>Bla bla bla <EM>evviva</EM> bla bla.</P>
```

54.4.3.1 Paragrafi e interruzioni

Si è già accennato al fatto che le righe vuote vengono ignorate in HTML. In effetti, l'interruzione di un paragrafo deve essere segnalata sempre esplicitamente, almeno attraverso l'indicazione dell'inizio di questo. Si osservi l'esempio seguente; anche se appare ovvio che il testo successivo alla dichiarazione del titolo è un paragrafo, questo modo non è ammissibile.

```
...
<H1>Titolo principale</H1>
Primo paragrafo che descrive qualcosa
che non serve precisare oltre.
<p>Paragrafo successivo.
<H1>Altro argomento</H1>
...
```

HTML ignora le righe bianche (possono contenere spazi e caratteri di tabulazione, oltre ai caratteri di conclusione della riga), per cui la separazione dei paragrafi attraverso l'inserzione di righe non serve a nulla.

Se si vuole ottenere l'interruzione della riga, in modo che il testo riprenda all'inizio, ma senza interrompere il paragrafo, potrebbe convenire l'utilizzo dell'elemento '**BR**', come nell'esempio seguente:

```
<P>Paragrafo che descrive qualcosa:<BR>
questa riga fa parte dello stesso paragrafo
che inizia con la riga precedente.</P>
```

Se si vogliono evitare i problemi causati dalle differenze nella composizione del risultato da parte dei programmi di navigazione, conviene ridurre al minimo l'uso di questo tipo di interruzione di riga.

Per separare il testo esiste anche la possibilità di utilizzare delle righe di separazione orizzontale: '**HR**' (*Horizontal rule*). Si tratta di elementi vuoti, per cui non si usa il marcatore di conclusione.

54.4.3.2 Elementi tipici utilizzati all'interno delle frasi

Nell'ambito del testo normale, si possono racchiudere alcune parti, per qualche motivo, all'interno di elementi specifici. Le situazioni tipiche riguardano l'evidenziamento, come nel caso degli elementi '**EM**' e '**STRONG**'.

```
... il <EM>codice di interruzione di riga</EM> è ciò
che separa le righe ...
```

La tabella 54.49 elenca gli elementi più comuni di questo tipo.

Tabella 54.49. Elementi da usare all'interno delle frasi.

Elemento	Significato
EM	Testo enfatizzato, di solito in corsivo.
STRONG	Testo evidenziato, di solito in neretto.
CITE	Citazione, nel senso di chi o cosa viene citato.
Q	Testo citato.
DFN	Definizione.
CODE	Codice usato in elaborazione, di solito reso in carattere dattilografico.
SAMP	Testo risultato di un'elaborazione.
KBD	Testo da inserire attraverso la tastiera.
VAR	Variabile o argomento di un programma.
ABBR	Abbreviazione.
ACRONYM	Acronimo.
SUB	Testo a pedice.
SUP	Testo ad apice.

Vale la pena di vedere come si può abbinare l'attributo '**TITLE**' agli elementi '**ABBR**' e '**ACRONYM**'. In generale, questi due si possono intendere quasi come la stessa cosa: spesso l'acronimo è un'abbreviazione. A parte il problema di scegliere questo o quell'elemento, l'attributo '**TITLE**' diventa utile per specificare il modo in cui si traduce l'acronimo o l'abbreviazione:

```
<ACRONYM TITLE="World Wide Web">WWW</ACRONYM>
```

A volte, un'abbreviazione o un acronimo diventano parole con un'identità propria; come tale acquisisce anche una pronuncia, che probabilmente si vuole preservare, specialmente quando il documento HTML viene letto attraverso un sistema vocale. Anche a questo scopo può essere usato l'attributo '**TITLE**'.

54.4.3.3 Citazioni

Il testo che si riferisce a una citazione si può delimitare attraverso due elementi: '**BLOCKQUOTE**' quando si tratta di blocchi di testo e '**Q**' quando si tratta di qualcosa che viene inserito nel flusso del testo normale.

```
<BLOCKQUOTE CITE="http://www.brot.dg/testi/prova.html"
 LANG="it">
<P>Bla bla bla...
bla bla bla bla...
bla bla bla.</P>
</BLOCKQUOTE>
```

Dall'esempio si può osservare l'utilizzo dell'attributo '**CITE**' con il quale si può indicare l'URI da dove ottenere il testo originale o il testo completo; inoltre è stato inserito anche l'attributo '**LANG**' allo scopo di specificare il linguaggio del testo citato (presumibilmente diverso da quello generale).

```
<P><CITE>Tizio Tizi</CITE> ha detto:
<Q CITE="http://www.brot.dg/testi/prova.html" LANG="it">
Bla bla bla...
bla bla bla bla...
bla bla bla.</Q></P>
```

Questo esempio ulteriore fa uso dell'elemento '**Q**', ma in aggiunta si vede anche l'elemento '**CITE**' con il quale viene indicato l'autore del testo citato.

54.4.3.4 Testo preformattato

In un documento HTML, l'unico modo per preservare gli spazi orizzontali e le interruzioni di riga, è l'uso dell'elemento '**PRE**'. In generale, il risultato che si ottiene viene rappresentato utilizzando un carattere dattilografico.

```
<P>Il comando <CODE>ls -l /</CODE> genera un risultato
simile a quello seguente:</P>

<PRE>
drwxr-xr-x    2 root     root         2048 gen  6 18:38 bin
drwxr-xr-x    3 root     root         1024 dic 31 08:08 boot
drwxr-xr-x    4 root     root        18432 gen 24 14:23 dev
drwxr-xr-x   68 root     root         4096 gen 24 14:09 etc
drwxr-sr-x   14 root     root         1024 gen  3 19:32 home
drwxr-xr-x    5 root     root         4096 gen  6 16:32 lib
drwxr-xr-x   19 root     root         1024 ago 15 16:02 mnt
drwxr-xr-x    5 root     root         1024 nov  9 14:59 opt
dr-xr-xr-x   88 root     root            0 gen 24 14:07 proc
drwxr-xr-x   18 root     root         1024 dic 16 17:37 root
drwxr-xr-x    3 root     root         2048 gen  6 16:12 sbin
drwxrwxrwt    6 root     root         8192 gen 24 18:56 tmp
drwxr-xr-x   16 root     root         1024 gen  5 15:23 usr
drwxr-xr-x   15 root     root         1024 set 29 15:02 var
</PRE>
```

Per essere sicuri del risultato finale, è bene evitare l'uso di caratteri di tabulazione, preferendo piuttosto gli spazi normali.

54.4.4 Elenchi

In generale, esistono tre tipi di elenchi: puntati, numerati e descrittivi. L'elenco puntato viene definito utilizzando l'elemento '**UL**' (*Unordered list*), quello numerato si ottiene con l'elemento '**OL**' (*Ordered list*), quello descrittivo si ottiene con l'elemento '**DL**' (*Definition list*). Le voci degli elenchi puntati e numerati sono costituite da elementi '**LI**' (*List item*), mentre nel caso dell'elenco descrittivo il contenuto si articola in elementi '**DT**' (*Definition term*) e '**DD**' (*Definition description*).

```
<UL>
    <LI>prima voce di un elenco puntato;</LI>
    <LI>seconda voce di un elenco puntato;</LI>
    <LI>terza voce.</LI>
</UL>
```

```
<OL>
    <LI>prima voce di un elenco numerato;</LI>
    <LI>seconda voce di un elenco numerato;</LI>
    <LI>terza voce.</LI>
</OL>
```

```
<DL>
    <DT>Resistenza</DT>
    <DT>Resistore</DT>
        <DD>Componente resistivo utilizzato in
            elettronica</DD>
    <DT>Condensatore</DT>
        <DD>Componente capacitivo...</DD>
</DL>
```

Gli esempi mostrano un uso molto semplice di questi elenchi. Si può osservare in particolare che nel caso dell'elenco descrittivo, gli elementi che delimitano il termine da descrivere possono essere più di uno simultaneamente.

Gli elementi '**LI**' e '**DT**' sono speciali, dal momento che possono contenere testo lineare normale, come si vede negli esempi, oppure dei blocchi di testo. Questo, tra le altre cose, consente di realizzare degli elenchi più complessi.

```
<OL>
    <LI><P>prima voce di un elenco numerato;</P></LI>
    <LI><P>seconda voce di un elenco numerato;</P></LI>
    <LI>
        <P>terza voce che si articola ulteriormente:</P>
        <UL>
            <LI>bla bla bla</LI>
            <LI>bla bla bla</LI>
            <LI>bla bla bla</LI>
        </UL>
    </LI>
</OL>
```

54.4.5 Tabelle

Quando si iniziano a utilizzare le tabelle e si scoprono gli effetti che si riescono a ottenere, non se ne vorrebbe più fare a meno. In realtà, sarebbe bene utilizzare le tabelle il meno possibile, perché alcuni programmi per la visualizzazione di documenti HTML non sono in grado di gestirle in maniera ottimale. Qui viene data solo una spiegazione superficiale, che comunque dovrebbe essere sufficiente per l'uso normale.

La tabella è definita dall'elemento '**TABLE**'; al suo interno può essere inclusa una didascalia rappresentata dall'elemento '**CAPTION**', quindi il contenuto della tabella viene distinto in intestazione, piede e corpo, all'interno dei quali si inseriscono le righe della tabella stessa (figura 54.58).

Figura 54.58. Esempio di una tabella.

```
.-------------------------------------------.   riga di
| Articolo       | Descrizione              |   intestazione
|================|==========================|
| 123xyz         | Bicicletta uomo          |  \
|----------------|--------------------------|  |
| 125xyz         | Bicicletta donna         |  >   corpo
|----------------|--------------------------|  |
| 121xyz         | Bicicletta bambino       |  /
|================|==========================|
| Articolo       | Descrizione              |      piede
'-------------------------------------------'
```

L'intestazione e il piede non sono obbligatori; in ogni caso, se si utilizzano vanno inseriti ordinatamente prima del corpo. Se non si indica l'intestazione o il piede, le righe che costituiscono il corpo hanno comunque bisogno di essere delimitate espressamente tra i marcatori che rappresentano l'elemento corrispondente.

Lo standard ISO 15445 obbliga all'utilizzo dell'attributo '**SUMMARY**' nell'elemento '**TABLE**'. Questo attributo dovrebbe permettere di riassumere il contenuto della tabella per quelle situazioni in cui potrebbe essere impossibile consultarla correttamente.

La tabella 54.59 riepiloga gli elementi utili nella realizzazione delle tabelle HTML.

Tabella 54.59. Elementi da usare per la realizzazione delle tabelle HTML.

Elemento	Significato
TABLE	Delimita la tabella.
CAPTION	Didascalia.
THEAD	Righe di intestazione.
TFOOT	Righe del piede.
TBODY	Righe del corpo.
TR	Riga normale.
TH	Elemento evidenziato di una riga.
TD	Elemento di una riga.

L'esempio seguente rappresenta una tabella molto banale, senza intestazione e senza piede:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

Il risultato è uno specchietto simile a quello che si vede di seguito:

```
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

Ecco lo stesso esempio con l'aggiunta di una riga di intestazione:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<THEAD>
    <TR><TD>Primo</TD><TD>Secondo</TD></TR>
</THEAD>
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

```
---------------------------------
 Primo          Secondo
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

L'esempio seguente aggiunge anche una didascalia molto breve:

```
<TABLE SUMMARY="uno due tre quattro cinque sei">
<CAPTION>
    Tabella banale
</CAPTION>
<THEAD>
    <TR><TD>Primo</TD><TD>Secondo</TD></TR>
</THEAD>
<TBODY>
    <TR><TD>uno</TD><TD>due</TD></TR>
    <TR><TD>tre</TD><TD>quattro</TD></TR>
    <TR><TD>cinque</TD><TD>sei</TD></TR>
</TBODY>
</TABLE>
```

```
        Tabella banale
---------------------------------
 Primo          Secondo
---------------------------------
 uno            due
 tre            quattro
 cinque         sei
---------------------------------
```

Le tabelle HTML possono essere molto più complesse di quanto è stato mostrato qui. Vale la pena di sottolineare il fatto che gli elementi '**TD**', ovvero le celle all'interno delle righe, possono contenere sia testo normale, sia blocchi di testo. Inoltre, lo standard ISO 15445

non consente più l'utilizzo di attributi per la descrizione dei bordi da far risaltare, perché per questo si possono applicare degli stili.

54.4.6 Riferimenti ipertestuali

La sigla HTML fa riferimento esplicitamente a un sistema ipertestuale. Ci deve quindi essere un modo per creare questi collegamenti.

Un riferimento può essere fatto a una pagina intera o a un punto particolare di una pagina. Il riferimento può essere assoluto, cioè provvisto dell'indicazione del nodo di rete e del percorso necessario a raggiungere la pagina, oppure può essere relativo al nodo attuale.

Per i riferimenti si utilizza l'elemento '**A**' ed eventualmente l'attributo '**ID**' di molti altri elementi.

54.4.6.1 Riferimenti a una pagina intera

Un riferimento a una pagina intera, con l'indicazione del percorso assoluto per raggiungerla, viene fatto come nell'esempio seguente:

```
<A HREF="http://www.brot.dg/prove/prova.html">Pagina di
 prova</A>
```

Nell'esempio, la frase «Pagina di prova» serve come punto di riferimento del puntatore a '`http://www.brot.dg/prove/prova.html`'.

Quando di realizza un documento HTML composto da più pagine collegate tra loro, è preferibile utilizzare riferimenti relativi, in modo da non dover indicare il nome del nodo in cui si trovano e nemmeno il percorso assoluto delle directory da attraversare per raggiungerle.

```
<A HREF="varie/nota.html">Annotazioni varie</A>
```

Nell'esempio, si vede un riferimento al file '`nota.html`' contenuto nella «directory» '`varie/`' discendente dalla directory corrente. La directory corrente, in questi casi, è quella in cui si trova la pagina contenente il puntatore.[1]

Il vantaggio di utilizzare riferimenti relativi, sta nella facilità con cui il documento può essere spostato o copiato in altri punti nel file system dello stesso o di un altro elaboratore (si veda anche quanto già scritto nella sezione 54.1).

54.4.6.2 Riferimenti a una posizione di una pagina

All'interno di una pagina è possibile collocare delle etichette che poi possono servire per fare dei riferimenti, sia a partire dalla stessa pagina che da altre. L'esempio seguente mostra un esempio di un'etichetta molto semplice.

```
<A NAME="introduzione"></A>
```

Si usa quindi lo stesso elemento che serve per creare un puntatore, ma con l'attributo '**NAME**'. L'argomento dell'attributo '**NAME**' (in questo caso è la parola '**introduzione**'), identifica quel punto.

Per fare riferimento a un'etichetta nella stessa pagina si può usare la forma dell'esempio seguente, con il quale si vuole puntare all'etichetta appena creata.

```
<A HREF="#introduzione">Introduzione</A>
```

Si utilizza l'attributo '**HREF**' come al solito, ma il suo argomento è il nome dell'etichetta preceduta dal simbolo '**#**'. Evidentemente, ciò è necessario per evitare di fare riferimento a un file con lo stesso nome.

Se si vuole fare riferimento a un'etichetta di un certo file, si utilizza la notazione solita, aggiungendo l'indicazione dell'etichetta.

```
<A HREF="http://www.brot.dg/varie/linux.html#introduzione">
Introduzione a GNU/Linux</A>
```

54.4.6.3 Collegamenti simmetrici

Si può osservare che l'elemento '**A**' serve sia per indicare un'etichetta, attraverso l'attributo '**NAME**', sia per definire un riferimento, attraverso l'attributo '**HREF**' (senza contare la possibilità di usare anche l'attributo '**ID**'). Questo fatto consente di realizzare dei riferimenti simmetrici, dove un riferimento è anche etichetta della terminazione opposta:

```
<A NAME="uno" HREF="#due">vai al punto due</A>
```

```
<A NAME="due" HREF="#uno">vai al punto uno</A>
```

L'esempio dovrebbe essere abbastanza chiaro: il primo puntatore punta al secondo, che a sua volta punta al primo.

54.4.6.4 Utilizzo dell'attributo ID

L'attributo '**ID**' è una generalizzazione attraverso la quale si attribuisce un'identità a un elemento. Può essere usato come destinazione per un riferimento fatto attraverso l'elemento '**A**' con l'attributo '**HREF**', ma il suo scopo è più ampio.

In generale, quando si realizzano dei riferimenti ipertestuali dovrebbe essere più conveniente l'indicazione di etichette attraverso l'attributo '**NAME**', dal momento che ci possono essere ancora dei navigatori o altri sistemi di lettura di file HTML che non sono in grado di riconoscere l'attributo '**ID**'.

54.4.7 Inserzioni di oggetti

Un documento HTML può contenere riferimenti a «oggetti» esterni. Nei casi più comuni si tratta di immagini o di applet, ma il concetto riguarda qualunque altra cosa che possa essere incorporata nel documento. Come si può supporre, l'elemento attraverso cui si includono gli oggetti è '**OBJECT**'. La tabella 54.73 elenca alcuni degli attributi di questo elemento.

Tabella 54.73. Alcuni attributi dell'elemento '**OBJECT**'.

Attributo	Significato
DATA	Riferimento al file dell'oggetto.
TYPE	Tipo di oggetto.
STANDBY	Messaggio di attesa durante il caricamento dell'oggetto.

Come si può intuire, il minimo per importare un oggetto richiede almeno l'uso dell'attributo '**DATA**'; inoltre, in generale è opportuno aggiungere anche l'attributo '**TYPE**' per precisare subito il tipo di oggetto.

L'elemento '**OBJECT**' non può essere vuoto; ciò che racchiude è quanto deve essere mostrato nel caso non sia possibile raggiungere l'oggetto indicato, oppure non sia possibile gestire l'oggetto stesso. Di solito si tratta di testo normale, ma potrebbe trattarsi di altri oggetti alternativi.

```
<OBJECT DATA="esempio.jpg" TYPE="image/jpg">Immagine di
esempio</OBJECT>
```

L'esempio mostra l'inclusione di un'immagine, '`esempio.jpg`', che nel caso non possa essere raggiunta o visualizzata, viene rimpiazzata con la frase: «Immagine di esempio». L'esempio seguente, al contrario, tenta di visualizzare un'altra immagine in un formato alternativo; se poi anche quella non è accessibile o visualizzabile, si passa al testo di prima:

```
<OBJECT DATA="esempio.png" TYPE="image/png">
    <OBJECT DATA="esempio.jpg" TYPE="image/jpg">
        Immagine di esempio
    </OBJECT>
</OBJECT>
```

54.4.7.1 Immagini

Il tipo di immagine che può essere visualizzata dipende solo dalle limitazioni del programma di navigazione o di composizione. Generalmente si possono utilizzare solo i formati GIF, JPG e PNG (in pratica le estensioni '`.gif`', '`.jpg`' e '`.png`').[2]

I riferimenti a file di immagine si fanno attraverso l'elemento '**OBJECT**' oppure '**IMG**'. In generale, per ottenere un documento HTML adatto alla maggior parte di programmi per la navigazione, conviene ancora utilizzare il vecchio elemento '**IMG**', come nell'esempio seguente:

```
<IMG SRC="http://www.brot.dg/varie/immagini/logo.jpg"
     ALT="Logo">
```

L'elemento '**IMG**' è vuoto, pertanto non si usa il marcatore di conclusione. Come si vede dall'esempio, si utilizza l'attributo '**SRC**' per definire la collocazione del file contenente l'immagine, l'attributo '**ALT**' per indicare una descrizione alternativa nel caso in cui l'immagine non possa essere visualizzata. La stessa cosa avrebbe potuto essere espressa con l'elemento '**OBJECT**' nel modo seguente:

```
<OBJECT DATA="http://www.brot.dg/varie/immagini/logo.jpg"
        TYPE="image/jpg">Logo</OBJECT>
```

Generalmente, per evitare problemi di compatibilità con i vari programmi di navigazione, è meglio evitare di fare scorrere il testo a fianco delle immagini, per cui è bene staccare il testo normale racchiudendolo esplicitamente all'interno di un elemento '**P**' (paragrafo).

```
<IMG SRC="immagini/logo.jpg" ALT="Logo">
<P>...testo che segue l'immagine...
```

L'immagine può essere utilizzata anche come pulsante per un riferimento ipertestuale, quando è contenuta all'interno di questo ultimo. In tali casi è particolarmente importante ricordare di inserire l'attributo '**ALT**', che diventa un'alternativa indispensabile nel caso in cui l'immagine non possa essere visualizzata.

```
<A HREF="varie/nota.html"><IMG SRC="img/nota.jpg"
ALT="Annotazioni varie"></A>
```

Naturalmente, se fosse necessario ricordarlo, non è obbligatorio che tutto si trovi sulla stessa riga, quindi l'esempio precedente può anche essere assemblato come indicato qui sotto:

```
<A HREF="varie/nota.html">
    <IMG SRC="immagini/nota.jpg" ALT="Annotazioni varie">
</A>
```

54.5 XHTML

XHTML è una rivisitazione dell'HTML in forma di applicazione XML, rimanendo molto simile all'HTML tradizionale, ma con alcune differenze importanti. In particolare:

- gli elementi devono essere delimitati correttamente con i marcatori di apertura e chiusura;
- non ci possono più essere elementi vuoti indicati con il solo marcatore di apertura, dal momento che al loro posto si possono solo usare i marcatori speciali nella forma '**<...>**';[3]
- i nomi degli elementi e degli attributi vanno scritti utilizzando solo lettere minuscole;
- gli attributi devono essere assegnati correttamente (non si possono usare più degli attributi booleani) e il valore assegnato deve essere delimitato da apici doppi o singoli;
- l'attributo '**lang**', se utilizzato, deve essere abbinato anche all'attributo '**xml:lang**', in base alle convenzioni dell'XML;
- se non si può evitare l'uso dell'attributo '**name**', questo deve essere abbinato anche all'attributo '**id**';
- se il valore assegnato a un attributo deve contenere una e-commerciale ('**&**'), occorre indicarla nella forma '**&**', anche se si tratta di un URI;
- se per qualche ragione non si dichiara la codifica utilizzata, deve trattarsi della forma UTF-8 oppure UTF-16;
- l'elemento '**isindex**' è superato e si preferisce usare l'elemento '**input**'.

54.5.1 Scheletro di un file XHTML

Trattandosi di un'applicazione XML, l'inizio dovrebbe essere scontato: si deve specificare che si tratta di un file XML, quindi si passa a indicare il DTD a cui si fa riferimento:


```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it"
      lang="it">
<head>
    <title>Esempio XHTML</title>
    <meta name="keywords" content="XML, SGML, XSL, DSSSL" />
</head>
<body>
    <p>Ciao mondo!</p>
</body>
</html>
```

L'esempio mostra un file XHTML completo, anche se molto breve. Si può osservare che il marcatore di apertura, oltre agli attributi '**xml:lang**' e '**lang**', contiene l'attributo '**xmlns**', a cui viene assegnato un URI prestabilito.

In XML, l'insieme di caratteri codificato è quello dell'insieme di caratteri universale. Di conseguenza, per la migliore compatibilità con il passato, la forma codificata del carattere più appropriata è UTF-8. Se il file utilizza l'ASCII tradizionale, senza estensioni, tutto va bene e non occorre altro; diversamente vanno usate preferibilmente le codifiche UTF-8 oppure UTF-16, come prevede in generale l'XML. La forma codificata del carattere viene specificata nell'istruzione iniziale, come già mostrato nell'esempio iniziale:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

54.5.2 Verifica della validità di un file XHTML

Volendo verificare la validità di un file XHTML attraverso il suo DTD, si può agire in modo simile a quanto si fa in generale con l'SGML. Questo è già descritto nella sezione 51.2; tuttavia occorre ricordare che la definizione SGML da utilizzare è quella specifica per l'XML.

Il DTD di XHTML e la definizione delle entità standard, possono essere ottenuti a partire da '`http://www.w3.org/TR/xhtml1/DTD/`' (ma probabilmente la directory non è leggibile), mentre la dichiarazione SGML si può trovare tra i sorgenti del pacchetto SP di James Clark. Si veda a questo proposito quanto già descritto nel capitolo 52.

54.6 CSS

I fogli di stile CSS (*Cascading style sheet*) rappresentano un metodo semplice per consentire di dichiarare e abbinare degli stili di composizione ai documenti HTML e ad altri tipi di sistemi SGML.

Il lavoro su CSS ha generato due «livelli», CSS1 e CSS2, intesi come la prima e la seconda versione del linguaggio di stile CSS. Teoricamente, il linguaggio CSS deve essere compatibile sia verso l'alto, sia verso il basso, nel senso che il primo livello CSS è compatibile con il secondo e il secondo è compatibile con il primo. In pratica, le estensioni fatte al linguaggio nel CSS2 sono tali per cui dovrebbero essere ignorate semplicemente dai programmi in grado di interpretare correttamente solo CSS1.

Nelle sezioni successive si introduce il linguaggio CSS affrontando solo parte delle caratteristiche del primo livello, con qualche annotazione eventuale sul secondo. Nella sezione 54.2.2 è descritto in quanti modi si può includere un foglio di stile CSS in un documento HTML, pertanto questo particolare non viene riproposto.

54.6.1 Logica del linguaggio CSS

Nella documentazione di CSS, le sue istruzioni vengono definite «regole», che si esprimono sinteticamente secondo la forma seguente, dove le parentesi graffe fanno parte della dichiarazione della regola:

selettore { ***dichiarazione*** }

Il principio è molto semplice: il «selettore» rappresenta qualcosa all'interno del documento; la dichiarazione è ciò che si vuole ottenere

su tale oggetto. All'interno di una regola si possono raggruppare più selettori, applicando così le stesse dichiarazioni; inoltre si possono indicare più dichiarazioni: i selettori si separano con la virgola; le dichiarazioni si separano con un punto e virgola:

```
selettore [, selettore]... { dichiarazione [; dichiarazione]... }
```

Le regole possono essere scritte anche utilizzando più righe di testo normale, per cui, la stessa sintassi appena mostrata potrebbe essere scritta anche come nel modo seguente (si osservi l'aggiunta di un punto e virgola ulteriore):

```
selettore [, selettore]... {
    dichiarazione ;
    [dichiarazione ;]
    ...
}
```

Teoricamente, quando si scrivono le regole iniziando ogni dichiarazione in una riga separata, è possibile evitare l'aggiunta del punto e virgola finale, ma questa scorciatoia non è consigliabile in generale.

Le dichiarazioni si scompongono a loro volta in proprietà e valori loro assegnati:

```
selettore [, selettore]... {
    proprietà : valore [valore_alternativo]... ;
    [proprietà : valore [valore_alternativo]... ;]
    ...
}
```

Come si vede, alle proprietà si possono assegnare più valori alternativi, in ordine di importanza.

Si osservi l'esempio seguente: attribuisce il colore blu al testo degli elementi '**H1**' di un documento HTML:

```
H1 { color: blue }
```

L'esempio successivo indica l'utilizzo di uno sfondo composto da un'immagine esterna per il corpo del documento, specificando che in mancanza dell'immagine, o in mancanza della possibilità di rappresentarla si può utilizzare uno sfondo bianco:

```
BODY { background: url(fondale.jpg) white }
```

Si intuisce che il nome del file contenente l'immagine è stato indicato come argomento di quello che sembra essere una funzione: ***url()***. Si osservi comunque che questa funzione fa riferimento a un URI e non a un URL, come fa intendere erroneamente il suo nome.

I commenti in un foglio di stile CSS si rappresentano in modo simile al linguaggio C, nella forma:

```
/* testo_ignorato */
```

54.6.1.1 Ereditarietà e collegamento in cascata

Una caratteristica fondamentale del linguaggio CSS è l'ereditarietà di talune caratteristiche in certe circostanze. Per comprendere il significato della cosa basta pensare alla struttura dell'HTML, o a un altro linguaggio SGML in generale: se si attribuisce una caratteristica stilistica a un elemento che per sua natura ne può contenere altri, ci si aspetta intuitivamente che questa si trasmetta anche ai livelli inferiori se applicabile, a meno che per tali elementi sia stato definito espressamente qualcosa di diverso.

Volendo fare un esempio più pratico, si può immaginare una caratteristica riferita alla dimensione del carattere di un blocco di testo. Se questo blocco contiene delle porzioni di testo delimitate da altri elementi, che possono servire per ottenere un testo enfatizzato in qualche modo, è normale attendersi che per queste porzioni venga utilizzata la stessa dimensione del carattere, senza bisogno di dichiarare esplicitamente e dettagliatamente questa richiesta.[4]

In generale, per quanto riguarda l'HTML, è normale assegnare all'elemento '**BODY**' le caratteristiche generali di tutto il documento, sfruttando il principio di ereditarietà.

L'altra caratteristica fondamentale del linguaggio CSS è la possibilità di definire gli stili in cascata. Questo significa che si possono abbinare assieme più fogli di stile e che nel complesso ci possono essere regole che si contraddicono a vicenda. Evidentemente, in questi casi viene applicato un criterio di scelta, che viene descritto più avanti.

54.6.1.2 Selettori

Il selettore di una regola CSS è qualcosa che rappresenta una parte del testo a cui si vogliono applicare le dichiarazioni relative. Nella situazione più semplice, il selettore viene indicato con il nome dell'elemento a cui si attribuisce. In questo modo, le dichiarazioni si applicano a tutti gli elementi di quel tipo. Nell'esempio seguente, che è già stato usato in precedenza, si attribuisce il colore blu al testo che compone tutti gli elementi '**H1**':

```
H1 { color: blue }
```

Tutti gli elementi HTML che si possono utilizzare nel corpo di tale tipo di documento possono utilizzare l'attributo '**CLASS**'. Questo permette di attribuire loro una «classe», ovvero un gruppo, di solito nell'ambito di quel tipo di elemento. Per indicare un selettore che faccia riferimento a una classe specifica di un certo elemento, si usa la notazione seguente:

```
[elemento] . classe
```

Come si vede, l'indicazione dell'elemento è facoltativa, in modo tale che, se non lo si indica, si faccia riferimento a tutti gli elementi che appartengono a quella stessa classe. L'esempio seguente mostra il caso degli elementi '**P**' che appartengono alla classe '**nota**', a cui viene abbinato il colore rosso per il testo:

```
P.nota { color: red }
```

L'esempio seguente mostra invece l'utilizzo di un selettore che fa riferimento a una classe di qualunque elemento:

```
.calmante { color: green }
```

Un selettore può essere anche più specifico e arrivare a individuare un elemento preciso nel documento HTML, attraverso il riferimento all'attributo '**ID**':

```
[elemento]#identificazione
```

In questa situazione non è necessario indicare il nome dell'elemento, dato che la stringa di identificazione è già un dato univoco per conto proprio. Al contrario, se si sbaglia l'indicazione dell'elemento, si annulla la validità della regola relativa, perché non può essere applicata. L'esempio seguente attribuisce all'elemento '**P**' identificato dalla stringa '**xyz**' il colore blu:

```
P#xyz { color: blu }
```

Un selettore può essere composto in modo da definire la dipendenza da un contesto. In altri termini, si può definire un selettore che dipende da un altro:

```
selettore sottoselettore [sotto_sottoselettore]...
```

Il primo selettore indica un ambito, all'interno del quale va cercata la corrispondenza per il secondo selettore, continuando eventualmente ad aumentare il dettaglio con altri selettori più specifici. Si osservi l'esempio seguente; serve a fare riferimento agli elementi '**EM**' che si trovano all'interno di un elemento '**H1**':

```
H1 EM { color: green }
```

È importante distinguere il raggruppamento di selettori dalla definizione di un contesto più dettagliato come in questo caso. Infatti, per

raggruppare i selettori si utilizza la virgola. L'esempio seguente applica il colore verde a tutti gli elementi '**EM**' contenuti all'interno di elementi '**H1**' o '**H2**':

```
H1 EM, H2 EM { color: green }
```

Un selettore può anche individuare una pseudo-classe, ovvero una zona di testo che viene individuata dal programma che si occupa di interpretare il documento HTML, che non corrisponde a elementi e classi indicati espressamente:

```
[elemento][.classe]:pseudo_classe
```

Il caso tipico di una pseudo-classe è quella che delimita la prima lettera di un elemento: '**first-letter**'. L'esempio seguente serve a ottenere una lettera iniziale più grande in tutti gli elementi '**P**' di classe '**primo**':

```
P.primo:first-letter {
    font-size: 200%;
    float: left;
}
```

54.6.1.3 Stili in cascata

I fogli di stile CSS possono essere uniti assieme in cascata. Tra le altre cose, ciò permette la definizione di uno o più stili da parte dell'autore e di uno o più stili personalizzati da parte dell'utente che legge il documento. Un file contenente lo stile CSS può incorporare altri file attraverso la direttiva '**@import**' che ha la sintassi seguente:

```
@import url(uri_foglio_di_stile);
```

Come si vede, riappare la funzione ***url()*** già mostrata in precedenza. In generale, le direttive di incorporazione dei fogli di stile esterni vanno collocate all'inizio del file, prima delle regole CSS.

Si è accennato al fatto che, nell'ambito dello stile complessivo che si ottiene, si possono generare dei conflitti tra dichiarazioni riferite alla stessa porzione di documento. Per scegliere quale dichiarazione deve avere la meglio, è necessario stabilire un peso differente, che dipende dal contesto e può anche essere condizionato attraverso l'aggiunta della stringa '**! important**' in coda alla dichiarazione:

```
H1 {
    color: black ! important;
    background: white ! important;
}
```

L'esempio mostra il caso in cui si tenta di aumentare il peso delle dichiarazioni che definiscono il colore del testo e dello sfondo negli elementi '**H1**'.

Viene descritta brevemente e in modo semplificato la sequenza attraverso cui vengono attribuite le caratteristiche dello stile.

- Le dichiarazioni vengono applicate se c'è la corrispondenza con i selettori. Se non ci sono corrispondenze, si applicano i valori ereditati; se non è possibile ereditare alcunché, si usano i valori iniziali.
- Le dichiarazioni vengono ordinate in base al loro peso, dove quelle marcate come «importanti» ricevono un peso maggiore rispetto a quelle normali.
- Le dichiarazioni vengono ordinate in base alla loro origine: lo stile dell'autore ha la precedenza su quello personalizzato dell'utente, che a sua volta ha la precedenza su quello predefinito dal programma utilizzato.
- Le dichiarazioni vengono ordinate in base alla precisione con cui individuano gli obiettivi. In pratica, le dichiarazioni più specifiche hanno la precedenza rispetto a quelle più generali.
- Al termine, se due regole hanno lo stesso peso, ha la precedenza quella che appare per ultima.

54.6.2 Proprietà

Di seguito vengono mostrate una serie di tabelle che descrivono l'utilizzo di alcune proprietà comuni nel linguaggio CSS. Bisogna ricordare che ogni programma di lettura o di composizione dei documenti HTML può fare la propria scelta su quali siano le dichiarazioni da prendere in considerazione, ignorando tutto il resto. Pertanto, si tratta solo di un'indicazione e l'utilizzo degli stili CSS deve essere sempre valutato tenendo conto delle carenze che poi ci possono essere in fase di lettura.

Tabella 54.93. Proprietà riferite ai caratteri.

Proprietà	Valori	Descrizione
`font-family`	*tipo_di_carattere*	Tipo di carattere.
`font-style`	`normal`	Forma normale.
	`italic`	Corsivo.
	`oblique`	Obliquo.
`font-variant`	`normal`	Serie normale.
	`small-caps`	Maiuscoletto.
`font-weight`	normal	Tono normale.
	`bold`	Nero.
	`bolder`	Nerissimo.
	`lighter`	Chiaro.
`font-size`	*n*`pt`	Dimensione in punti.
	n`cm`	Dimensione in centimetri.
	n`mm`	Dimensione in millimetri.
	n`em`	Dimensione relativa in quadratoni.
	n`ex`	Dimensione relativa in Ex.
	n`%`	Dimensione relativa percentuale.
	`small`	Carattere piccolo.
	`medium`	Carattere normale.
	`large`	Carattere grande.

Nella tabella 54.93 si fa riferimento in particolare alla proprietà '**font-family**'. A questa può essere attribuito il nome di una famiglia di caratteri, oppure il nome di una «famiglia generica», che in pratica identifica uno stile del carattere senza indicare esattamente quale tipo di carattere. Una famiglia di caratteri potrebbe essere '**times**', mentre una famiglia generica potrebbe essere '**serif**', ovvero un carattere munito di grazie. Alla proprietà '**font-family**' possono essere abbinati più tipi di caratteri, separati da una virgola, per indicare una sequenza alternativa da utilizzare in mancanza di quello preferito:

```
BODY { font-family: gill, helvetica, sans-serif }
```

L'esempio mostra proprio questo: prima si tenta di utilizzare il carattere '**gill**'; quindi si prova con '**helvetica**'; infine ci si accontenta di un carattere senza grazie, '**sans-serif**'.

Tabella 54.95. Proprietà riferite ai colori e allo sfondo.

Proprietà	Valori	Descrizione
`color`	*colore*	Colore del carattere o di primo piano.
`background-color`	*colore*	Colore dello sfondo.
`background-image`	`url(`*uri*`)`	Immagine da usare per lo sfondo.

Per quanto riguarda i colori (tabella 54.95), si possono indicare attraverso il nome che questi hanno in inglese, oppure attraverso la funzione ***rgb()***, con la quale si specifica il valore RGB:

```
rgb(livello_rosso, livello_verde, livello_blu)
```

I numeri che esprimono i livelli dei colori fondamentali RGB vanno da 0 a 255.

Tabella 54.96. Proprietà riferite al testo.

Proprietà	Valori	Descrizione
`vertical-align`	`baseline`	Testo al livello normale.
	`middle`	Allinea al centro.
	`sub`	Pedice.
	`super`	Apice.
`text-transform`	`none`	Nessuna trasformazione del testo.
	`capitalize`	Rende maiuscola la prima lettera delle parole.
	`uppercase`	Tutto maiuscolo.
	`lowercase`	Tutto minuscolo.
`text-align`	`left`	Allinea a sinistra.
	`right`	Allinea a destra.
	`center`	Centra.
	`justify`	Allinea a sinistra e a destra.
`text-indent`	*n*`pt`	Rientro in punti.
	n`cm`	Rientro in centimetri.
	n`mm`	Rientro in millimetri.
	n`em`	Rientro relativo in quadratoni.
	n`ex`	Rientro relativo in Ex.
	n`%`	Rientro relativo in percentuale.
`line-height`	`normal`	Altezza normale della riga.
	n`pt`	Altezza in punti.
	n`cm`	Altezza in centimetri.
	n`mm`	Altezza in millimetri.
	n`%`	Altezza relativa in percentuale.

Tabella 54.97. Proprietà riferite al testo racchiuso in blocchi rettangolari.

Proprietà	Valori	Descrizione
`margin-top`	`auto`	Margine superiore automatico.
	n`pt`	Margine superiore in punti.
	n`cm`	Margine superiore in centimetri.
	n`mm`	Margine superiore in millimetri.
	n`%`	Margine superiore relativo in percentuale.
`margin-bottom`	`auto`	Margine inferiore automatico.
	n`pt`	Margine inferiore in punti.
	n`cm`	Margine inferiore in centimetri.
	n`mm`	Margine inferiore in millimetri.
	n`%`	Margine inferiore relativo in percentuale.
`margin-left`	`auto`	Margine sinistro automatico.
	n`pt`	Margine sinistro in punti.
	n`cm`	Margine sinistro in centimetri.
	n`mm`	Margine sinistro in millimetri.
	n`%`	Margine sinistro relativo in percentuale.
`margin-right`	`auto`	Margine destro automatico.
	n`pt`	Margine destro in punti.
	n`cm`	Margine destro in centimetri.
	n`mm`	Margine destro in millimetri.
	n`%`	Margine destro relativo in percentuale.
`border-width`	`thin`	Bordo sottile.
	`medium`	Bordo medio.
	`thick`	Bordo spesso.
`border-color`	***colore***	Colore del bordo.
`border-style`	`none`	Bordo non visibile.
	`dotted`	Bordo puntato.
	`dashed`	Bordo tratteggiato.
	`solid`	Bordo continuo.
	`double`	Bordo continuo doppio.
`width`	`auto`	Larghezza automatica.
	n`pt`	Larghezza in punti.
	n`cm`	Larghezza in centimetri.
	n`mm`	Larghezza in millimetri.
	n`%`	Larghezza relativa in percentuale.
`height`	`auto`	Altezza automatica.
	n`pt`	Altezza in punti.
	n`cm`	Altezza in centimetri.
	n`mm`	Altezza in millimetri.
	n`%`	Altezza relativa in percentuale.
`float`	`none`	Posizione fissa.
	`left`	A sinistra con testo che scorre a destra.
	`right`	A destra con testo che scorre a sinistra.
`clear`	`none`	Scorre normalmente.
	`left`	Salta un oggetto che si trova a sinistra.
	`right`	Salta un oggetto che si trova a destra.
	`both`	Salta qualunque oggetto fluttuante.

54.6.3 Definizione della pagina

Il secondo livello del linguaggio CSS, introduce una regola speciale, '**@page**', per la definizione della pagina, nel momento in cui il documento dovesse essere stampato. Inoltre, sono disponibili delle proprietà specifiche per l'impaginazione da usarsi nelle regole normali. In generale, la regola '**@page**' viene usata per definire i margini ed eventualmente anche le dimensioni della pagina. L'esempio seguente dichiara una pagina A4 utilizzando margini tutti uguali di 2 cm:

```
@page {
  size 210mm 297mm;
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 2cm;
  margin-right: 2cm;
}
```

La stessa cosa si potrebbe ottenere in modo meno dettagliato come segue:

```
@page {
  size 210mm 297mm;
  margin: 2cm;
}
```

La tabella 54.100 riepiloga le proprietà più importanti riferite a questa regola.

Tabella 54.100. Proprietà riferite alla regola speciale '**@page**'.

Proprietà	Valori	Descrizione
`size`	*x y*	Ampiezza e altezza della pagina (nelle varie unità di misura).
	`auto`	Definisce le dimensioni e l'orientamento in modo automatico.
	`landscape`	Orientamento orizzontale.
	`portrait`	Orientamento verticale.
`margin`	*x*	Dimensione di tutti i margini.
	n`pt`	Dimensione in punti.
	n`cm`	Dimensione in centimetri.
	n`mm`	Dimensione in millimetri.
	n`%`	Dimensione relativa in percentuale.
`margin-left`	*x*	Dimensione del margine sinistro.
`margin-right`	*x*	Dimensione del margine destro.
`margin-top`	*x*	Dimensione superiore.
`margin-bottom`	*x*	Dimensione inferiore.

La regola '**@page**' può essere usata in modo da distinguere tra pagine destre e pagine sinistre. Si osservi a questo proposito l'esempio seguente:

```
@page :left {
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 4cm;
  margin-right: 2cm;
}

@page :right {
  margin-top: 2cm;
  margin-bottom: 2cm;
  margin-left: 2cm;
  margin-right: 4cm;
}
```

Come accennato sono disponibili delle proprietà specifiche per l'impaginazione da usarsi nelle regole normali. Con queste si intende controllare la suddivisione del testo in pagine, imponendo un salto pagina, oppure impedendolo nell'ambito dell'elemento coinvolto.

Queste proprietà non vengono descritte qui, ma è utile almeno tenere in considerazione la loro esistenza.

54.7 JavaScript

Il linguaggio JavaScript somiglia al Java, semplificato in modo da poter essere eseguito direttamente, come uno script comune. Questo linguaggio è però destinato a essere interpretato da un navigatore, inserito normalmente all'interno di documenti HTML, pertanto il suo ambito di utilizzo è più limitato rispetto a Java.

Oltre alla limitatezza del contesto a cui è destinato questo linguaggio di programmazione, occorre considerare il fatto che, essendo interpretato dal navigatore, può esistere solo una compatibilità di massima, perché tutto dipende dalla capacità del navigatore stesso di eseguire le istruzioni del linguaggio.

Si intende che il ruolo del linguaggio JavaScript è quello di arricchire di potenzialità i documenti ipertestuali che si possono leggere attraverso un navigatore, ma in generale il suo utilizzo va ponderato, per non escludere quella porzione di utenti che si trova a usare software che per qualche ragione non può eseguire script di questo tipo.

Prima di poter affrontare la descrizione del linguaggio, è necessario comprendere come si usa in pratica, dal momento che ciò richiede l'inserimento, in qualche modo, all'interno di documenti ipertestuali. Si osservi l'esempio seguente che mostra il contenuto di un file HTML molto semplice:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 1</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
           document.write ("Ciao a tutti! ");
           document.write ("Questo &egrave; il mio primo ");
           document.write ("programma JavaScript.");
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 1</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

Se si legge questo file con un navigatore comune, si dovrebbe ottenere di visualizzare questo risultato:

Ciao a tutti! Questo è il mio primo programma JavaScript.

JavaScript 1

Bla bla bla bla...

Per cominciare si deve osservare che è stato utilizzato l'elemento '**SCRIPT**' all'interno dell'intestazione del file. Attraverso l'attributo '**LANGUAGE**' è stato specificato l'uso del linguaggio JavaScript e con l'attributo '**TYPE**' (che in questo caso è facoltativo) viene specificato il tipo MIME.

Il contenuto dell'elemento '**SCRIPT**' è il programma JavaScript che, come si vede dal risultato, viene eseguito prima di analizzare il corpo della pagina HTML. Per garantire di non ottenere effetti spiacevoli in presenza di un navigatore che non conosce l'uso di questo elemento, convenzionalmente, si delimita il codice inserito attraverso un commento particolare:

```
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            …
            …
```

```
            ...
        //-->
    </SCRIPT>
```

Si osservi che la chiusura del commento deve essere realizzata necessariamente secondo la forma '`//-->`', per garantire la massima compatibilità con i navigatori.

In alternativa, si può ottenere lo stesso risultato scrivendo il codice JavaScript in un file separato:

```
document.write ("Ciao a tutti! ");
document.write ("Questo &egrave; il mio primo ");
document.write ("programma JavaScript.");
```

Supponendo che questo file si chiami '`javascript-001.js`', il file HTML, collocato nella stessa directory, potrebbe essere realizzato semplicemente così:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 1</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript"
        SRC="javascript-001.js"></SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 1</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

Il risultato che si ottiene è lo stesso.

Negli esempi che vengono mostrati qui a proposito delle pagine HTML che incorporano o fanno riferimento a programmi JavaScript, si specifica in modo esplicito il tipo di standard HTML usato e si cerca di scrivere in modo corretto, secondo la sintassi prevista. Tuttavia, nessuno degli standard HTML, tanto meno lo standard ISO che viene usato proprio negli esempi, prevede l'inserzione o il riferimento a programmi JavaScript nel modo mostrato. Pertanto, questi file sono errati formalmente, per ciò che riguarda il linguaggio HTML, ma rappresentano l'unico modo per garantire un discreto grado di compatibilità tra i navigatori che devono eseguire gli script contenuti.

Negli esempi mostrati appare un solo elemento '**`SCRIPT`**' nell'intestazione del file HTML. In pratica ci possono essere più elementi '**`SCRIPT`**' nell'intestazione e ne possono apparire anche nel corpo del documento, come nell'esempio seguente:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 2</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Ciao a tutti! ");
        //-->
    </SCRIPT>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Questo &egrave; il mio ");
            document.write ("secondo programma ");
            document.write ("JavaScript.");
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 2</H1>
<P>Bla bla bla bla...</P>

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
```



```
        document.write ("Come si pu&ograve; vedere, ");
        document.write ("&egrave; possibile inserire ");
        document.write ("del codice JavaScript ");
        document.write ("anche nel corpo del file HTML.");
    //-->
</SCRIPT>
</BODY>
</HTML>
```

Ecco il risultato che si dovrebbe ottenere:

Ciao a tutti! Questo è il mio secondo programma JavaScript.

JavaScript 2

Bla bla bla bla...

Come si può vedere, è possibile inserire del codice JavaScript anche nel corpo del file HTML.

Gli esempi mostrati fino a questo punto fanno uso esclusivo della funzione ***document.write()***, ovvero, più precisamente, del metodo ***write()*** dell'oggetto '**`document`**'. Ciò che si ottiene utilizzandolo è di inserire la stringa che viene data come argomento nel documento che viene visualizzato. Pertanto, quanto viene inserito nell'intestazione appare all'inizio, mentre ciò che viene inserito nel corpo, appare in corrispondenza dell'inserzione.

Dal momento che il documento in questione è scritto con il linguaggio HTML, queste stringhe devono essere coerenti con il linguaggio HTML stesso, cosa che negli esempi mostrati non è stato, a parte l'uso di alcune macro per le vocali accentate. L'esempio appena presentato, andrebbe modificato, come appare qui di seguito:

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 3</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("<P>Ciao a tutti! ");
        //-->
    </SCRIPT>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Questo &egrave; il mio ");
            document.write ("secondo programma ");
            document.write ("JavaScript.</P>");
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 3</H1>
<P>Bla bla bla bla...</P>

<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        document.write ("<P>Come si pu&ograve; vedere, ");
        document.write ("&egrave; possibile inserire del ");
        document.write ("codice JavaScript anche nel ");
        document.write ("corpo del file HTML.</P>");
    //-->
</SCRIPT>
</BODY>
</HTML>
```

In pratica, viene circoscritto il testo all'interno di un elemento '**`P`**'. Naturalmente, nello stesso modo si può inserire del codice HTML più complesso, anche se per evitare di fare troppa confusione, sarebbe meglio ridurre al minimo questa possibilità.

Generalmente, quando si scrive un programma JavaScript non si può disporre di un analizzatore ortografico per la ricerca di errori, anche se banali. In tal modo, l'unica cosa su cui si può contare è il programma che lo interpreta, ovvero il navigatore, che normalmente si limita a non eseguire il programma che non sia sintatticamente perfetto. Pertanto, ciò significa che la scrittura di programmi complessi diventa un compito molto difficile e a volte impossibile.

54.7.1 Verifica sintattica

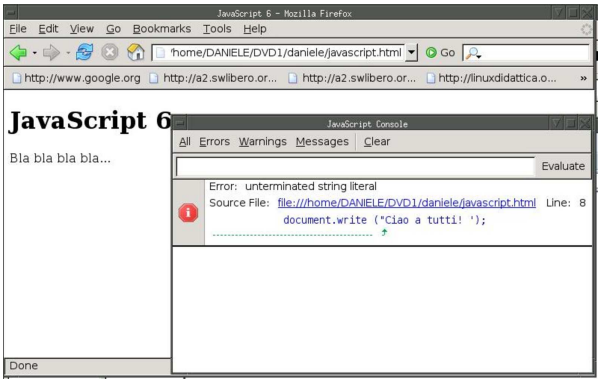
Al programmatore JavaScript manca un analizzatore sintattico standardizzato universale e il controllo si può eseguire solo attraverso gli strumenti di questo o di quel navigatore. Fortunatamente, Mozilla (Netscape) e molti dei suoi vari derivati, offre la «console JavaScript», dove vengono messi in evidenza quelli che possono essere degli errori del programma.

Viene proposto un esempio HTML contenente un programma JavaScript con diversi errori.

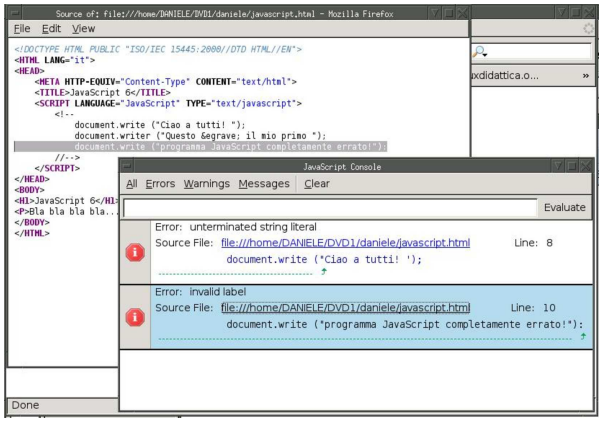
Inizialmente viene visualizzato il file con Mozilla Firefox e viene aperta la console JavaScript selezionando la voce *JavaScript Console* dal menù *Tools*.

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>JavaScript 6</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
        <!--
            document.write ("Ciao a tutti! ');
            document.writer ("Questo &egrave; il mio ");
            document.writer ("primo programma JavaScript ");
            document.write ("completamente errato!"):
        //-->
    </SCRIPT>
</HEAD>
<BODY>
<H1>JavaScript 6</H1>
<P>Bla bla bla bla...</P>
</BODY>
</HTML>
```

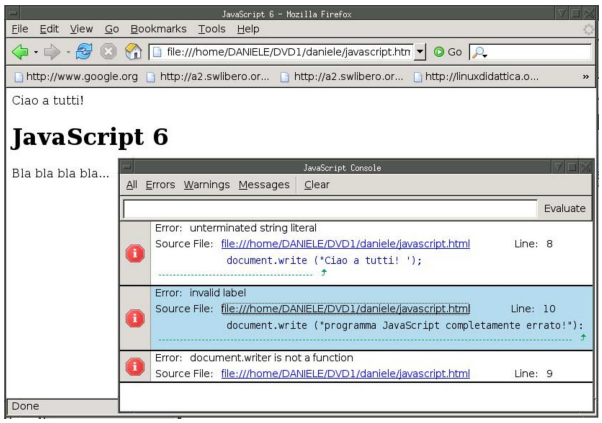
Inizialmente viene visualizzato il file con Mozilla Firefox e viene aperta la console JavaScript selezionando la voce *JavaScript Console* dal menù *Tools*:



Come si vede dalla figura, il primo errore a essere segnalato è la stringa terminata in modo errato. Si ipotizza di correggere l'errore e di ricaricare la pagina:



In questo caso, seguendo il collegamento visibile in corrispondenza dell'errore che viene segnalato, si vede apparire una finestra contenente il sorgente con la riga errata evidenziata. L'errore che viene messo in risalto è dovuto alla conclusione dell'istruzione con i due punti, invece che con il punto e virgola; si ipotizza di correggere anche questo errore e di ricaricare la pagina:



L'ultimo errore a essere notato è il nome di una funzione inesistente: '`writer`'.

Altri navigatori derivati da Mozilla offrono una console simile, eventualmente da voci diverse del menù.

54.7.2 Caratteristiche generali del linguaggio di programmazione

Il linguaggio JavaScript è molto vicino a Java, con delle semplificazioni significative. Le istruzioni hanno la forma del linguaggio C, pertanto terminano con un punto e virgola ('`;`'); inoltre i raggruppamenti si ottengono con le parentesi graffe ('`{ }`'). I commenti ricalcano la forma usata per il linguaggio C:

```
/* commento_generico */
```

```
// commento_fino_alla_fine_della_riga
```

La gestione delle variabili di JavaScript è semplificata, al punto che non esiste la dichiarazione del tipo ed eventualmente la conversione avviene in modo automatico (salva la necessità di eseguire delle conversioni specifiche, quando quelle automatiche non avvengono nel modo desiderato).

Il punto critico della compatibilità dei programmi realizzati in JavaScript sta nell'utilizzo di oggetti (con le proprietà e i metodi relativi) che sono prestabiliti e creati automaticamente. Evidentemente, la disponibilità di questi oggetti dipende dal programma usato per interpretare il linguaggio JavaScript, ovvero dal navigatore; pertanto si può contare solo su un numero limitato di questi, così come bi-

sogna essere prudenti nell'uso delle proprietà e dei metodi relativi, anche se ci possono essere dei navigatori o altri interpreti in grado di accettare l'utilizzo di oggetti, istanze e metodi molto sofisticati.

Teoricamente è possibile creare degli oggetti nuovi, ma in pratica si tratta di un procedimento sconsigliabile, pertanto conviene limitarsi all'uso di quelli predefiniti, creando eventualmente delle funzioni personalizzate.

54.7.3 Variabili, costanti, tipi di dati ed espressioni

Le variabili possono essere dichiarate implicitamente, nel momento in cui vengono utilizzate, oppure si può usare una forma esplicita, ma in ogni caso non viene specificato il tipo del loro contenuto:

```
{var} nome{ = valore_assegnato};
```

Pertanto, la variabile ***x*** può essere dichiarata implicitamente così:

```
x = 1;
```

Oppure, si può usare la dichiarazione esplicita:

```
var x = 1;
```

I nomi delle variabili possono essere composti da lettere e numeri, che devono iniziare necessariamente con una lettera. Volendo seguire la convenzione del linguaggio Java, i nomi delle variabili si possono comporre unendo assieme una serie di parole che consentono di intendere il ruolo delle variabili stesse, utilizzando l'iniziale maiuscola per ogni parola che compone l'insieme del nome, tranne che per la prima, che inizia senza iniziale maiuscola. Per esempio: '**`miaStringa`**', '**`elencoNomiUtenti`**', '**`indiceElenco`**'.

I tipi di dati principali sono molto pochi, dal momento che nella gestione dei numeri si distingue soltanto tra interi e numeri con virgola mobile.

Tabella 54.116. Tipi di dati principali e rappresentazione delle costanti.

Tipo	Esempi di costante
intero	'**`0`**', '**`123`**', '**`45678`**'
virgola mobile	'**`0.1`**', '**`123.45`**', '**`45678.901`**'
booleano	'**`true`**', '**`false`**'
stringa	'**`"bla bla bla"`**', '**`"0422.1234567"`**', '**`'bene bene'`**'

Dagli esempi mostrati nella tabella riepilogativa dei tipi principali, si può osservare che i valori numerici in virgola mobile utilizzano il punto per separare la parte intera da quella decimale; inoltre, le stringhe sono delimitate indifferentemente con apici doppi oppure singoli. Nelle tabelle successive si riepilogano gli operatori principali delle espressioni che si possono realizzare con JavaScript.

Tabella 54.117. Elenco degli operatori aritmetici e di quelli di assegnamento relativi a valori numerici.

Operatore e operandi	Descrizione
`++`***op***	Incrementa di un'unità l'operando prima che venga restituito il suo valore.
op`++`	Incrementa di un'unità l'operando dopo averne restituito il suo valore.
`--`***op***	Decrementa di un'unità l'operando prima che venga restituito il suo valore.
op`--`	Decrementa di un'unità l'operando dopo averne restituito il suo valore.
`+`***op***	Non ha alcun effetto.
`-`***op***	Inverte il segno dell'operando.
op1 `+` ***op2***	Somma i due operandi.
op1 `-` ***op2***	Sottrae dal primo il secondo operando.
op1 `*` ***op2***	Moltiplica i due operandi.

Operatore e operandi	Descrizione
op1 `/` ***op2***	Divide il primo operando per il secondo.
op1 `%` ***op2***	Modulo -- il resto della divisione tra il primo e il secondo operando.
var `=` ***valore***	Assegna alla variabile il valore alla destra.
op1 `+=` ***op2***	***op1*** `=` ***op1*** `+` ***op2***
op1 `-=` ***op2***	***op1*** `=` ***op1*** `-` ***op2***
op1 `*=` ***op2***	***op1*** `=` ***op1*** `*` ***op2***
op1 `/=` ***op2***	***op1*** `=` ***op1*** `/` ***op2***
op1 `%=` ***op2***	***op1*** `=` ***op1*** `%` ***op2***

Tabella 54.118. Elenco degli operatori di confronto. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
op1 `==` ***op2***	*Vero* se gli operandi si equivalgono.
op1 `!=` ***op2***	*Vero* se gli operandi sono differenti.
op1 `<` ***op2***	*Vero* se il primo operando è minore del secondo.
op1 `>` ***op2***	*Vero* se il primo operando è maggiore del secondo.
op1 `<=` ***op2***	*Vero* se il primo operando è minore o uguale al secondo.
op1 `>=` ***op2***	*Vero* se il primo operando è maggiore o uguale al secondo.

Tabella 54.119. Elenco degli operatori logici. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
`!` ***op***	Inverte il risultato logico dell'operando.
op1 `&&` ***op2***	Se il risultato del primo operando è *Falso* non valuta il secondo.
op1 `||` ***op2***	Se il risultato del primo operando è *Vero* non valuta il secondo.

Tabella 54.120. Concatenamento di stringhe. Le metavariabili indicate rappresentano gli operandi e la loro posizione.

Operatore e operandi	Descrizione
stringa `+` ***op***	Crea una stringa unica concatenando la stringa a sinistra con il valore a destra, previa conversione di tipo, se necessaria.
op `+` ***stringa***	Crea una stringa unica concatenando il valore a sinistra, previa conversione di tipo, se necessaria, assieme alla stringa che si trova alla destra.

54.7.4 Funzioni e campo di azione delle variabili

Come già accennato nel capitolo, il linguaggio JavaScript offre oggetti già pronti, dei quali si usano le proprietà e i metodi relativi (ma con prudenza, per motivi di compatibilità), quindi consente di realizzare delle funzioni, che si dichiarano in modo molto semplice:

```
function nome ([parametro[, parametro]…]) {
    istruzione
    …
}
```

Come si può intendere, nel modello mostrato le parentesi graffe fanno parte della dichiarazione della funzione e servono a raggruppare le istruzioni che questa contiene.

Una funzione può restituire un valore; in tal caso si usa l'istruzione '**return**':

```
return valore;
```

Le variabili dichiarate all'interno della funzione hanno una campo di azione locale e oscurano temporaneamente variabili globali, con lo stesso nome, dichiarate al di fuori della funzione. Le variabili dichiarate al di fuori delle funzioni, ma prima della loro chiamata, sono accessibili all'interno delle stesse, se non sono oscurate.

54.7.5 Strutture di controllo di flusso

Le strutture di controllo di flusso disponibili nel linguaggio JavaScript sono sostanzialmente le stesse del linguaggio Java, tenendo conto però che alcuni navigatori potrebbero non riconoscere le strutture più sofisticate. Vengono mostrati i modelli sintattici relativi alle strutture più comuni e più compatibili, con qualche esempio e poche spiegazioni.

- ```
  if (condizione) istruzione
  ```

  ```
  if (condizione) istruzione else istruzione
  ```

  Se la condizione si verifica, viene eseguita l'istruzione (o il gruppo di istruzioni) seguente; quindi il controllo passa alle istruzioni successive alla struttura. Se viene utilizzato '**else**', nel caso non si verifichi la condizione, viene eseguita l'istruzione che ne segue. Vengono mostrati alcuni esempi:

  ```
  var importo;
  …
  if (importo > 10000000) document.write ("Offerta vantaggiosa");
  ```

  ```
  var importo;
  var memorizza;
  …
  if (importo > 10000000)
    {
      memorizza = importo;
      document.write ("Offerta vantaggiosa");
    }
  else
    {
      document.write ("Non conviene");
    }
  ```

  ```
  var importo;
  var memorizza;
  …
  if (importo > 10000000)
    {
      memorizza = importo;
      document.write ("Offerta vantaggiosa");
    }
  else if (importo > 5000000)
    {
      memorizza = importo;
      document.write ("Offerta accettabile");
    }
  else
    {
      document.write ("Non conviene");
    }
  ```

- ```
  while (condizione) istruzione
  ```

  Il comando '**while**' esegue un'istruzione, o un gruppo di queste, finché la condizione continua a restituire il valore *Vero*. La condizione viene valutata prima di eseguire il gruppo di istruzioni e poi ogni volta che termina un ciclo, prima dell'esecuzione del successivo. L'esempio seguente visualizza 10 volte la lettera «x»:


```
var contatore = 0;
while (contatore < 10)
  {
    contatore++;
    document.write ("x");
  }
```

Nel blocco di istruzioni di un ciclo '**while**', ne possono apparire alcune particolari:

– '**break**'
  esce definitivamente dal ciclo '**while**';
– '**continue**'
  interrompe l'esecuzione del gruppo di istruzioni e riprende dalla valutazione della condizione.

L'esempio seguente è una variante del ciclo di visualizzazione mostrato sopra, modificato in modo da vedere il funzionamento di '**break**'. Si osservi che '**while (true)**' equivale a un ciclo senza fine, perché la condizione è sempre vera:

```
var contatore = 0;

while (true)
  {
    if (contatore >= 10)
      {
        break;
      }
    contatore++;
    document.write ("x");
  }
```

- ```
  do blocco_di_istruzioni while (condizione);
  ```

  Il comando '**do**' esegue un gruppo di istruzioni una volta e poi ne ripete l'esecuzione finché la condizione continua a restituire il valore *Vero*.

- ```
  for (espressione1; espressione2; espressione3) istruzione
  ```

  Questa è la forma tipica di un'istruzione '**for**', in cui la prima espressione corrisponde all'assegnamento iniziale di una variabile, la seconda a una condizione che deve verificarsi fino a che si vuole che sia eseguita l'istruzione (o il gruppo di istruzioni), mentre la terza serve per l'incremento o decremento della variabile inizializzata con la prima espressione. In pratica, potrebbe esprimersi nella sintassi seguente:

  ```
  for (var = n; condizione; var++) istruzione
  ```

  Il ciclo '**for**' potrebbe essere definito anche in maniera differente, più generale: la prima espressione viene eseguita una volta sola all'inizio del ciclo; la seconda viene valutata all'inizio di ogni ciclo e il gruppo di istruzioni viene eseguito solo se il risultato è *Vero*; l'ultima viene eseguita alla fine dell'esecuzione del gruppo di istruzioni, prima che si ricominci con l'analisi della condizione.

  L'esempio in cui viene visualizzata per 10 volte una «x», potrebbe tradursi nel modo seguente, attraverso l'uso di un ciclo '**for**':

  ```
  var contatore;

  for (contatore = 0; contatore < 10; contatore++)
    {
      document.write ("x");
    }
  ```

54.7.6 Array

A differenza degli altri tipi di dati mostrati, le variabili che devono fare riferimento a un array devono essere dichiarate come tali:

```
var nome = new Array();
```

Si osservi che le variabili dichiarate in questo modo **fanno riferimento** a un array, pertanto non contengono direttamente l'array stesso. Ciò significa che, se il nome di un array viene indicato in una chiamata di funzione, se la funzione modifica il contenuto dell'array, la modifica riguarda lo stesso array a cui fa riferimento la variabile nella chiamata.

Gli elementi di un array dichiarato in questo modo vengono creati automaticamente nel momento in cui sono utilizzati, come si vede nell'esempio seguente:

```
var mioArray = new Array();
//
mioArray[0] = "Ciao";
mioArray[1] = "come";
mioArray[7] = "stai";
//
document.write (mioArray[0]);
document.write (" ");
document.write (mioArray[1]);
document.write (" ");
document.write (mioArray[7]);
```

In questo caso sono stati saltati volutamente gli elementi dall'indice due all'indice sei, che esistono, ma contenenti un valore indefinito.

Nel linguaggio JavaScript, così come in Java, gli array sono degli oggetti e come tali possono disporre di proprietà e di metodi. Quando gli elementi dell'array vengono dichiarati in questo modo, cioè con un indice numerico intero, è possibile leggere il metodo '**`length`**', che consente di conoscere la lunghezza dell'array stesso:

```
var mioArray = new Array();
//
mioArray[0] = "Ciao";
mioArray[1] = "come";
mioArray[7] = "stai";
//
document.write (mioArray.length);
```

In questo caso, il valore che viene visualizzato attraverso ***document.write()*** è il numero otto, dal momento che array contiene, formalmente, otto elementi.

L'indice dell'array può essere indicato attraverso una stringa, come nell'esempio seguente:

```
var mioArray = new Array();
//
mioArray["a"] = "Ciao";
mioArray["s"] = "come";
mioArray["d"] = "stai";
//
document.write (mioArray["a"]);
document.write (" ");
document.write (mioArray["s"]);
document.write (" ");
document.write (mioArray["d"]);
```

Questi indici sono o diventano dei metodi dell'array, consentendo una forma più semplice per indicare i vari elementi:

```
var mioArray = new Array();
//
mioArray["a"] = "Ciao";
mioArray.s    = "come";
mioArray.d    = "stai";
//
document.write (mioArray.a);
document.write (" ");
document.write (mioArray["s"]);
document.write (" ");
document.write (mioArray["d"]);
```

In pratica, indicare '**`mioArray["a"]`**', oppure '**`mioArray.a`**', dovrebbe essere la stessa cosa.

La proprietà '**`length`**' dovrebbe funzionare solo per gli elementi numerati in modo tradizionale, ignorando quelli a cui si accede per mezzo di un indice differente.

54.7.7 Funzioni standard

Alcune «funzioni» standard facilitano l'uso del linguaggio. Nelle tabelle successive si riepilogano quelle più comuni che dovrebbero essere compatibili nella maggior parte delle situazioni.

Tabella 54.131. Funzioni generali, ovvero metodi che non richiedono l'indicazione di un oggetto.

Funzione	Descrizione
`escape(`***stringa***`)`	Restituisce la stringa indicata come argomento, dopo averla trasformata in modo da poterla rappresentare in un indirizzo URI. La trasformazione implica l'uso delle sequenze formate con il prefisso '**`%`**'.
`unescape(`***stringa***`)`	Restituisce la stringa indicata come argomento, dopo averla trasformata in modo da tradurre le sequenze formate con il prefisso '**`%`**' nei caratteri che queste rappresentano. In pratica è l'inverso di ***escape()***.
`parseFloat(`***stringa***`)`	Elabora la stringa restituendo il numero a virgola mobile che questa rappresenta (o che dovrebbe rappresentare).
`parseInt(`***stringa***`)`	Elabora la stringa restituendo il numero intero che questa rappresenta (o che dovrebbe rappresentare).
`eval(`***stringa***`)`	Esegue il contenuto della stringa come un'istruzione e restituisce il valore che questa istruzione restituisce a sua volta.
`isNaN(`***valore***`)`	Verifica un valore e restituisce *Vero* se questo risulta essere indefinito.

Tabella 54.132. Metodi matematici, che si applicano necessariamente all'oggetto '**`Math`**', come mostrato in modo esplicito nel modello sintattico.

Metodo	Descrizione
`Math.abs(`***n***`)`	Restituisce il valore assoluto di ***n***
`Math.ceil(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per eccesso.
`Math.round(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per eccesso se ***n*** $\geq 0{,}5$, altrimenti arrotondato per difetto.
`Math.floor(`***n***`)`	Restituisce il valore intero di ***n*** arrotondato per difetto.
`Math.exp(`***n***`)`	Restituisce e^{n} (dove «e» approssima il valore 2,718).
`Math.log(`***n***`)`	Restituisce il logaritmo naturale di ***n***.
`Math.max(`***n_1, n_2, …n_n***`)`	Restituisce il massimo tra i valori numerici indicati.
`Math.min(`***n_1, n_2, …n_n***`)`	Restituisce il minimo tra i valori numerici indicati.
`Math.pow(`***x, y***`)`	Restituisce il risultato di x^{y}.
`Math.random()`	Restituisce un numero casuale, a virgola mobile, nell'intervallo compreso tra zero e uno.
`Math.sqrt(`***n***`)`	Restituisce la radice quadrata di ***n***.
`Math.sin(`***n***`)`	Restituisce il seno di ***n***.
`Math.cos(`***n***`)`	Restituisce il coseno di ***n***.
`Math.tan(`***n***`)`	Restituisce la tangente di ***n***.
`Math.asin(`***n***`)`	Restituisce l'arcoseno di ***n***.
`Math.acos(`***n***`)`	Restituisce l'arcocoseno di ***n***.
`Math.atan(`***n***`)`	Restituisce l'arcotangente di ***n***.

Tabella 54.133. Istruzioni per la creazione di oggetti contenenti informazioni data-orario.

Istruzione	Descrizione
`x = new Date()`	Costruisce l'oggetto ***x*** contenente la data e l'orario corrente.
`x = new Date(anno, mese, giorno)` `x = new Date(anno, mese, giorno, ore, minuti, secondi)`	Costruisce l'oggetto ***x*** contenente la data ed eventualmente anche l'orario indicati.

Tabella 54.134. Metodi per leggere o modificare informazioni data-orario a oggetti creati con ***Date()***.

Metodo	Descrizione
`x.setYear(anno)`	Imposta l'anno nell'oggetto ***x***.
`x.setMonth(anno)`	Imposta il mese nell'oggetto ***x***.
`x.setDate(giorno)`	Imposta il giorno del mese nell'oggetto ***x***.
`x.setHours(ora)`	Imposta l'ora nell'oggetto ***x***.
`x.setMinutes(minuti)`	Imposta i minuti nell'oggetto ***x***.
`x.setSeconds(secondi)`	Imposta i secondi nell'oggetto ***x***.
`x.getYear()`	Restituisce l'anno contenuto nell'oggetto ***x***.
`x.getMonth()`	Restituisce il mese contenuto nell'oggetto ***x***.
`x.getDay()`	Restituisce il giorno contenuto nell'oggetto ***x***.
`x.getHours()`	Restituisce l'ora contenuta nell'oggetto ***x***.
`x.getMinutes()`	Restituisce i minuti contenuti nell'oggetto ***x***.
`x.getSeconds()`	Restituisce i secondi contenuti nell'oggetto ***x***.

Si osservi che l'anno potrebbe essere restituito in forma non corretta; per esempio, l'anno 2004 potrebbe risultare come il numero 104; nello stesso modo, l'anno 1904 potrebbe apparire come il numero 4.

Tabella 54.135. Funzioni varie.

Funzione	Descrizione
`alert(stringa)`	Mostra la stringa all'interno di una finestra di avvertimento.
`setTimeout(stringa, n)`	Esegue la funzione scritta nella stringa che rappresenta il primo parametro, dopo aver atteso ***n*** ms (millesimi di secondo).

54.7.8 Gestione delle stringhe

Anche per JavaScript, come per Java, le stringhe sono oggetti, per i quali esistono metodi appropriati ed esiste anche la proprietà '**`length`**', che restituisce la lunghezza della stringa (come già avviene per gli array). Molti dei metodi riferiti alle stringhe servono per delimitare il testo con dei marcatori appropriati, per esempio per ottenere un testo nero o un corsivo, o ancora qualcosa di più appariscente.

Tabella 54.136. Metodi relativi alla gestione delle stringhe.

Metodo	Descrizione
`x.charAt(n)`	Restituisce un carattere nella posizione ***n*** della stringa contenuta nell'oggetto ***x***, dove il primo carattere viene identificato con lo zero.
`x.substring(n, m)`	Restituisce la sottostringa ottenuta a partire alla posizione ***n*** fino alla posizione ***m*** esclusa.

Metodo	Descrizione
`x.indexOf(stringa[, n])`	Restituisce la posizione a partire dalla quale la stringa indicata come primo parametro corrisponde con la stringa contenuta nell'oggetto ***x***. Se è indicato anche il secondo parametro, la ricerca parte dalla posizione ***n***. Se la corrispondenza non viene trovata, restituisce il valore `-1`.
`x.lastIndexOf(stringa[, n])`	Restituisce la posizione più a destra dove la stringa indicata come primo parametro corrisponde con la stringa contenuta nell'oggetto ***x***. Se è indicato anche il secondo parametro, la ricerca termina in corrispondenza dalla posizione ***n***. Se la corrispondenza non viene trovata, restituisce il valore `-1`.
`x.toLowerCase()`	Restituisce la stringa contenuta nell'oggetto ***x*** convertendola in lettere minuscole.
`x.toUpperCase()`	Restituisce la stringa contenuta nell'oggetto ***x*** convertendola in lettere maiuscole.
`x.fontcolor(colore)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<font color="`*colore*`">`**' e '**`</font>`**' all'inizio e alla fine della stessa.
`x.fontsize(dimensione)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<font size="`*dimensione*`">`**' e '**`</font>`**' all'inizio e alla fine della stessa.
`x.blink()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<blink>`**' e '**`</blink>`**' all'inizio e alla fine della stessa.
`x.bold()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<b>`**' e '**`</b>`**' all'inizio e alla fine della stessa.
`x.italics()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<i>`**' e '**`</i>`**' all'inizio e alla fine della stessa.
`x.fixed()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<tt>`**' e '**`</tt>`**' all'inizio e alla fine della stessa.
`x.big()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<big>`**' e '**`</big>`**' all'inizio e alla fine della stessa.
`x.small()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<small>`**' e '**`</small>`**' all'inizio e alla fine della stessa.
`x.sub()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<sub>`**' e '**`</sub>`**' all'inizio e alla fine della stessa.
`x.sup()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<sup>`**' e '**`</sup>`**' all'inizio e alla fine della stessa.
`x.strike()`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<strike>`**' e '**`</strike>`**' all'inizio e alla fine della stessa.
`x.anchor(nome)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<a name="`*nome*`">`**' e '**`</a>`**' all'inizio e alla fine della stessa.

Metodo	Descrizione
`x.link(`*riferimento*`)`	Restituisce la stringa contenuta nell'oggetto ***x***, aggiungendo i marcatori '**`<a href="`*riferimento*`">`**' e '**`</a>`**' all'inizio e alla fine della stessa.

54.7.9 Moduli «FORM»

I dati che vengono inseriti all'interno di moduli HTML (elementi '**`FORM`**'), possono essere letti come proprietà di oggetti che discendono da '**`document`**'. All'interno di un elemento '**`FORM`**', si inseriscono normalmente una serie di elementi '**`INPUT`**', con caratteristiche differenti; sia l'elemento '**`FORM`**', sia l'elemento '**`INPUT`**' hanno la possibilità di avere un nome, attraverso l'attributo '**`NAME`**'. Nella situazione più semplice, si può avere un elemento '**`INPUT`**' per l'inserimento manuale di un testo da parte dell'utente, come nell'esempio seguente, dove per il momento l'attributo '**`NOME`**' dell'elemento '**`FORM`**' non sarebbe indispensabile:

```
<FORM NAME="elaborazione" METHOD="get"
 ACTION="/cgi-bin/elabora">
<P><INPUT NAME="nominativo" TYPE="TEXT" SIZE="30">
<INPUT TYPE="submit" VALUE="Invia"></P>
</FORM>
```

Un programma JavaScript ha la possibilità di leggere il testo inserito nel campo '**`nominativo`**' attraverso la proprietà '**`document.elaborazione.nominativo.value`**'; in pratica, sono stati usati i valori degli attributi '**`NAME`**' per costruire l'oggetto relativo. Per esempio, si potrebbe modificare l'esempio nel modo seguente:

```
<FORM METHOD="get" ACTION="/cgi-bin/elabora">
<P><INPUT TYPE="TEXT" SIZE="30" NAME="nominativo">
<INPUT TYPE="button" VALUE="Invia"
 onClick="verificaInvia()"></P>
</FORM>
```

Come si vede, il bottone finale è stato modificato, da un tipo '**`submit`**' a un tipo '**`button`**', per evitare che l'informazione venga inviata con la sola selezione del bottone, che invece provoca l'esecuzione della funzione ***verificaInvia()***, che intuitivamente ha lo scopo di verificare i dati inseriti e di inviarli se corretti. La funzione di controllo potrebbe essere realizzata così:

```
function verificaInvia () {
    if (document.elaborazione.nominativo.value == "")
      {
        alert ("È necessario inserire il nominativo!");
        document.elaborazione.nominativo.focus();
      }
    else
      {
        document.elaborazione.submit();
      }
}
```

Intuitivamente, osservando l'esempio, si comprende che se il campo risulta vuoto nel momento in cui si seleziona il bottone, viene mostrato un avvertimento e viene messo a fuoco proprio il campo da correggere, altrimenti viene inviato il contenuto del modulo '**`elaborazione`**'.

In considerazione del fatto che non si può avere la certezza di avere un utenza che può disporre in ogni circostanza di un navigatore compatibile con il programma JavaScript che si scrive, si può essere più morbidi e dare la possibilità di inviare i dati senza alcun controllo:

```
<FORM METHOD="get" ACTION="/cgi-bin/elabora">
<P><INPUT TYPE="TEXT" SIZE="30" NAME="nominativo">
<INPUT TYPE="button" VALUE="Invia"
 onClick="verificaInvia()"></P>
<INPUT TYPE="submit" VALUE="Invia senza controllo"></P>
</FORM>
```

Nelle tabelle successive vengono descritti alcuni eventi, metodi e proprietà relative all'uso di elementi '**`FORM`**'.

Tabella 54.141. Eventi relativi all'elemento '**`FORM`**' e all'elemento '**`INPUT`**'.

Modello	Descrizione
`<FORM … onSubmit="`*funzione*`">`	Esegue la funzione quando il modulo viene inviato.
`<FORM … onReset="`*funzione*`">`	Esegue la funzione quando il modulo viene azzerato.
`<INPUT … onFocus="`*funzione*`">`	Esegue la funzione quando il componente è a fuoco.
`<INPUT … onBlur="`*funzione*`">`	Esegue la funzione quando il componente non è più a fuoco.
`<INPUT … onClick="`*funzione*`">`	Esegue la funzione quando il componente viene selezionato con un clic.
`<INPUT … onSelect="`*funzione*`">`	Esegue la funzione quando viene selezionata una porzione del testo del componente.

Tabella 54.142. Alcune proprietà e metodi riferiti a oggetti derivanti da elementi '**`FORM`**' e '**`INPUT`**'. La metavariabile ***x*** rappresenta il nome associato all'elemento '**`FORM`**' e ***y*** rappresenta il nome associato all'elemento '**`INPUT`**'.

Modello	Descrizione
`document.`***x***`.action`	Rappresenta l'indirizzo usato normalmente nell'attributo '**`ACTION`**' dell'elemento '**`FORM`**'.
`document.`***x***`.encoding`	Codifica usata nel modulo.
`document.`***x***`.method`	Modalità di invio dei dati del modulo ('**`GET`**' o '**`POST`**').
`document.`***x***`.submit()`	Invia i dati del modulo.
`document.`***x***`.reset()`	Azzera il contenuto del modulo.
`document.`***x***`.`***y***`.type`	Restituisce il tipo di campo.
`document.`***x***`.`***y***`.value`	Restituisce il valore del campo.
`document.`***x***`.`***y***`.focus()`	Seleziona l'elemento (lo mette a fuoco).
`document.`***x***`.`***y***`.blur()`	Deseleziona l'elemento (l'opposto della messa a fuoco).

Dovendo lavorare con nomi di proprietà e metodi molto lunghi, può essere conveniente dichiarare il contesto con l'istruzione '**`with`**':

```
with (prefisso) {
    …
}
```

L'esempio già apparso della funzione ***verificaInvia()*** potrebbe essere semplificato agli occhi del lettore modificandolo così:

```
function verificaInvia () {
    with (document.elaborazione)
      {
        if (nominativo.value == "")
          {
            alert ("È necessario inserire il nominativo!");
            nominativo.focus();
          }
        else
          {
            submit();
          }
      }
}
```

54.7.10 Esempi di programmazione

Si tenga in considerazione il fatto che in condizioni normali non si ha la disponibilità di un analizzatore sintattico del linguaggio JavaScript, pertanto la ricerca di errori banali di sintassi diventa problematica, dal momento che il navigatore comune si limita a non eseguire il programma che gli risulta non essere corretto. Anche per questa ragione non vengono mostrati esempi di una certa complessità.

54.7.10.1 Fattoriale

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Calcola il fattoriale di un valore intero</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function fattoriale (x)
        {
            var i = x - 1;
            //
            while (i > 0)
              {
                x = x * i;
                i--;
              }
            return x;
        }
        //
        function esegui ()
        {
            var x = parseInt (document.modulo.x.value);
            var z;
            var risultato = "";
            z = fattoriale (x);
            //
            risultato = risultato
                      + x
                      + "!"
                      + "="
                      + z;
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Calcola il fattoriale di un valore intero</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

In alternativa, l'algoritmo si può tradurre in modo ricorsivo:

```
function fattoriale (x)
{
    if (x > 1)
      {
        return (x * fattoriale (x - 1));
      }
    else
      {
        return 1;
      }
    //
    // Teoricamente non dovrebbe arrivare qui.
    //
}
```

54.7.10.2 Massimo comune divisore

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">

    <TITLE>Determina il massimo comune divisore tra due
           numeri interi positivi</TITLE>

    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function mcd (x, y)
        {
            var i;
            var z = 0;
            //
            while (x != y)
              {
                if (x > y)
                  {
                    x = x - y;
                  }
                else
                  {
                    y = y - x;
                  }
              }
            return x;
        }
        //
        function esegui ()
        {
            var x = parseInt (document.modulo.z.value);
            var y = parseInt (document.modulo.y.value);
            var z;
            var risultato = "";
            z = mcd (x, y);
            //
            risultato = risultato
                      + "mcd("
                      + x
                      + ", "
                      + y
                      + ")"
                      + "="
                      + z;
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Determina il massimo comune divisore tra due numeri
interi positivi</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x"><BR>
    y = <INPUT TYPE="TEXT" SIZE="30" NAME="y">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

54.7.10.3 Numero primo

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Determina se un numero intero sia primo o
    meno</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function primo (x)
        {
            var primo = true;
            var i = 2;
            var j;
            //
            while ((i < x) && primo)
            {
                j = x / i;
                j = Math.floor (j);
                j = x - (j * i);
                //
                if (j == 0)
                  {
                    primo = false;
```

```
                }
              else
                {
                  i++;
                }
          }
          return primo;
      }
      //
      function esegui ()
      {
          var x = parseInt (document.modulo.x.value);
          var z;
          var risultato = "";
          z = primo (x);
          //
          if (z)
            {
              risultato = risultato
                    + x
                    + " è un numero primo!";
            }
          else
            {
              risultato = risultato
                    + x
                    + " non è un numero primo!";
            }
          alert (risultato);
      }
      //-->
      </SCRIPT>
</HEAD>
<BODY>
<H1>Determina se un numero intero sia primo o meno</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x">
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

54.7.10.4 Ricerca sequenziale

```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Ricerca sequenziale</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        function ricercaseq (lista, x, a, z)
        {
            var i;
            //
            // Scandisce l'array alla ricerca dell'elemento.
            //
            for (i = a; i <= z; i++)
              {
                if (x == lista[i])
                  {
                    return i;
                  }
              }
            //
            // La corrispondenza non è stata trovata.
            //
            return -1;
        }
        function esegui ()
        {
            var a = new Array ();
            var z;
            var risultato = "";
            var x = document.modulo.x.value;
            //
            a[0] = document.modulo.a0.value;
            a[1] = document.modulo.a1.value;
            a[2] = document.modulo.a2.value;
            a[3] = document.modulo.a3.value;
            a[4] = document.modulo.a4.value;
            a[5] = document.modulo.a5.value;
            a[6] = document.modulo.a6.value;
            a[7] = document.modulo.a7.value;
            a[8] = document.modulo.a8.value;
            a[9] = document.modulo.a9.value;
            //
            z = ricercaseq (a, x, 0, (a.length-1));
            //
            risultato = x
                      + " si trova nella posizione "
                      + z
                      + ".";
            alert (risultato);
        }
        //-->
        </SCRIPT>
</HEAD>
<BODY>
<H1>Ricerca sequenziale</H1>

<FORM NAME="modulo" METHOD="get" ACTION="">
<P>
    x = <INPUT TYPE="TEXT" SIZE="30" NAME="x"><BR>
    a[0] = <INPUT TYPE="TEXT" SIZE="30" NAME="a0"><BR>
    a[1] = <INPUT TYPE="TEXT" SIZE="30" NAME="a1"><BR>
    a[2] = <INPUT TYPE="TEXT" SIZE="30" NAME="a2"><BR>
    a[3] = <INPUT TYPE="TEXT" SIZE="30" NAME="a3"><BR>
    a[4] = <INPUT TYPE="TEXT" SIZE="30" NAME="a4"><BR>
    a[5] = <INPUT TYPE="TEXT" SIZE="30" NAME="a5"><BR>
    a[6] = <INPUT TYPE="TEXT" SIZE="30" NAME="a6"><BR>
    a[7] = <INPUT TYPE="TEXT" SIZE="30" NAME="a7"><BR>
    a[8] = <INPUT TYPE="TEXT" SIZE="30" NAME="a8"><BR>
    a[9] = <INPUT TYPE="TEXT" SIZE="30" NAME="a9"><BR>
</P>
<P>
    <INPUT TYPE="button" VALUE="esegui" onClick="esegui()">
</P>
</FORM>

</BODY>
</HTML>
```

Esiste anche una soluzione ricorsiva che viene mostrata nella subroutine seguente:

```
function ricercaseq (lista, x, a, z)
{
    if (a > z)
      {
        //
        // La corrispondenza non è stata trovata.
        //
        return -1;
      }
    else if (x == lista[a])
      {
        return a;
      }
    else
      {
        return ricercaseq (lista, x, a+1, z);
      }
}
```

54.8 Approfondimento: verifiche automatiche con JavaScript

Questa sezione contiene la descrizione di un programma molto semplice, scritto con il linguaggio JavaScript, per la realizzazione di verifiche che generano automaticamente la valutazione. Il programma riguarda una verifica di esempio e serve solo come idea di un modello da utilizzare per la realizzazione di altre verifiche del genere. Si osservi che un meccanismo molto simile a quello descritto qui si

può utilizzare all'interno di Alml, il quale genera automaticamente il codice JavaScript necessario.

Viene mostrato subito il listato completo della pagina HTML contenente il programma; nelle sezioni successive viene descritto nelle sue varie componenti.

```
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html">
    <TITLE>Verifica banale</TITLE>
    <SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
    <!--
        //
        // Something to do at the beginning.
        //
        // Get the initial time stamp.
        //
        var start_time = new Date ();
        //
        // Countdown function.
        //
        function countdown (time_allowed)
        {
            var hours;
            var minutes;
            var seconds;
            var time;
            var time_string;
            var time_sign;
            //
            // Verify the countdown time sign.
            //
            if (time_allowed > 0)
              {
                time_sign = "";
              }
            else
              {
                time_sign = "-";
              }
            //
            // Find the absolute countdown value.
            //
            time = Math.abs (time_allowed);
            //
            hours   = (time - (time % 3600000)) / 3600000;
            time    = time - (hours * 3600000);
            minutes = (time - (time % 60000)) / 60000;
            time    = time - (minutes * 60000);
            seconds = (time - (time % 1000)) / 1000;
            //
            if (minutes < 10)
              {
                minutes = "0" + minutes;
              }
            if (seconds < 10)
              {
                seconds = "0" + seconds;
              }
            if (hours < 10)
              {
                hours = "0" + hours;
              }
            //
            time_string = time_sign + hours + ":" + minutes
                          + ":" + seconds;
            //
            // Show the time string inside some text field.
            //
            document.test_form_1.countdown_0.value
             = time_string;
            document.test_form_1.countdown_1.value
             = time_string;
            document.test_form_1.countdown_2.value
             = time_string;
            //
            // Reduce 10 s inside the countdown variable.
            //
            time_allowed = time_allowed - 10000;
            //
            // Wait for 10 s for the recursive call.
            //
            setTimeout ("countdown (" + time_allowed + ")",
                        10000);
            //
        }
        //
        // Radio button scan function.
        //
        function scan_radio_button (radio_button)
        {
            var counter;
            //
            for (counter = 0;
                 counter < radio_button.length;
                 counter++)
              {
                if (radio_button[counter].checked)
                  {
                    return (counter + 1);
                  }
              }
            //
            // Nothing found.
            //
            return 0;
        }
        //
        // Check test_form_1.
        //
        function check_test_form_1 (start_time)
        {
            //
            var end_time = new Date ();
            var test_time = end_time - start_time;
            var time_allowed = 1 * 60000;
            var time_delay = test_time - time_allowed;
            //
            if (time_delay < 0)
              {
                time_delay = 0;
              }
            //
            // Penalty for a minute delay.
            //
            var score_time_penalty = 0.5;
            //
            // Set the result for every choice.
            //
            var score_dita_di_una_mano = new Array();
            score_dita_di_una_mano[0] = 0;
            score_dita_di_una_mano[1] = 0;
            score_dita_di_una_mano[2] = 0;
            score_dita_di_una_mano[3] = 0;
            score_dita_di_una_mano[4] = 0;
            score_dita_di_una_mano[5] = 0;
            score_dita_di_una_mano[6] = 5;
            score_dita_di_una_mano[7] = 0;
            score_dita_di_una_mano[8] = 0;
            //
            // Set the result for every choice.
            //
            var score_dita_di_un_piede = new Array();
            score_dita_di_un_piede[0] = 0;
            score_dita_di_un_piede[1] = 0;
            score_dita_di_un_piede[2] = 0;
            score_dita_di_un_piede[3] = 0;
            score_dita_di_un_piede[4] = 0;
            score_dita_di_un_piede[5] = 0;
            score_dita_di_un_piede[6] = 5;
            score_dita_di_un_piede[7] = 0;
            score_dita_di_un_piede[8] = 0;
            //
            // Save the personal data.
            //
            var student
                = document.test_form_1.student.value;
            var date = document.test_form_1.date.value;
            //
            // Scan the radio buttons.
            //
            var dita_di_una_mano
```

```
            = scan_radio_button
                (document.test_form_1.dita_di_una_mano);
        //
        var dita_di_un_piede
            = scan_radio_button
                (document.test_form_1.dita_di_un_piede);
        //
        // Calculate the results.
        //
        var account_score_time_penalty;
        var account_score_dita_di_una_mano;
        var account_score_dita_di_un_piede;
        var account_score_sum = 0;
        //
        account_score_time_penalty
            = (time_delay / 1000)
               * (score_time_penalty / 60);
        //
        account_score_dita_di_una_mano
         = score_dita_di_una_mano[dita_di_una_mano];
        account_score_dita_di_un_piede
         = score_dita_di_un_piede[dita_di_un_piede];
        account_score_sum
         = account_score_sum
            - account_score_time_penalty;
        account_score_sum
         = account_score_sum
            + account_score_dita_di_una_mano;
        account_score_sum
         = account_score_sum
            + account_score_dita_di_un_piede;
        //
        // Reset the form to avoid the use of the [Back]
        // button from the browser.
        //
        document.test_form_1.reset();
        //
        // Print a simple HTML header.
        //
        document.writeln ("<HTML>");
        document.writeln ("<HEAD>");
        document.write   ("<TITLE>");
        document.write   ("Risultato della verifica");
        document.writeln ("</TITLE>");
        document.writeln ("</HEAD>");
        document.writeln ("<BODY>");
        //
        // Print a H1 title and some personal data.
        //
        document.writeln ("<H1>Verifica banale</H1>");
        document.writeln ("<P>studente: " + student
                          + "</P>");
        document.writeln ("<P>data: " + date + "</P>");
        //
        // Print about time.
        //
        document.write ("<P>inizio della verifica: ");
        document.write (start_time.getYear());
        document.write (".");
        document.write (start_time.getMonth() + 1);
        document.write (".");
        document.write (start_time.getDate());
        document.write (" ");
        document.write (start_time.getHours());
        document.write (":");
        document.write (start_time.getMinutes());
        document.write (".");
        document.write (start_time.getSeconds());
        document.writeln ("</P>");
        //
        document.write ("<P>conclusione della ");
        document.write ("verifica: ");
        document.write (end_time.getYear());
        document.write (".");
        document.write (end_time.getMonth() + 1);
        document.write (".");
        document.write (end_time.getDate());
        document.write (" ");
        document.write (end_time.getHours());
        document.write (":");
        document.write (end_time.getMinutes());
```



```
        document.write (".");
        document.write (end_time.getSeconds());
        document.writeln ("</P>");
        //
        document.write ("<P>durata complessiva della ");
        document.write ("verifica: ");
        document.write (test_time / 1000);
        document.write (" s, ");
        document.write (test_time / 1000 / 60);
        document.write (" m;");
        document.writeln ("</P>");
        //
        document.write ("<P>tempo a disposizione: ");
        document.write (time_allowed / 1000);
        document.write (" s; ");
        document.write ("ritardo: ");
        document.write (time_delay / 1000);
        document.write (" s; ");
        document.write ("penalit&agrave;/minuto: ");
        document.write (score_time_penalty);
        document.write ("; ");
        document.write ("penalit&agrave; ");
        document.write ("complessiva: ");
        document.write (account_score_time_penalty);
        document.writeln ("</P>");
        //
        // First question.
        //
        document.write ("<P>Quante sono le dita ");
        document.write ("di una mano? risposta n. ");
        document.write (dita_di_una_mano);
        document.write ("; ");
        document.write ("punteggio: ");
        document.write (account_score_dita_di_una_mano);
        document.writeln ("</P>");
        //
        // Second question.
        //
        document.write ("<P>Quante sono le dita ");
        document.write ("di un piede? risposta n. ");
        document.write (dita_di_un_piede);
        document.write ("; ");
        document.write ("punteggio: ");
        document.write (account_score_dita_di_un_piede);
        document.writeln ("</P>");
        //
        // Final result.
        //
        document.write ("<P>Punteggio complessivo ");
        document.write ("della verifica: ");
        document.write (account_score_sum);
        document.writeln ("</P>");
        //
        // End of HTML.
        //
        document.writeln ("</BODY>");
        document.writeln ("</HTML>");
        //
        // Print with a printer.
        //
        window.print ();
        //
        // Stop loading.
        //
        window.stop ();
    }
    //-->
    </SCRIPT>
</HEAD>
<BODY onLoad="document.test_form_1.reset();
              countdown (1 * 60000);">

<H1>Verifica banale</H1>

<FORM NAME="test_form_1" METHOD="get" ACTION="">

    <P>Tempo a disposizione: <INPUT TYPE="TEXT"
    NAME="countdown_0" SIZE="10" VALUE=" "></P>

    <P>Cognome, nome, classe:
    <INPUT TYPE="TEXT" SIZE="60" NAME="student" VALUE="">
```

```
    </P>

    <P>Data:
    <INPUT TYPE="TEXT" SIZE="60" NAME="date" VALUE="">
    </P>

    <P>Tempo a disposizione: <INPUT TYPE="TEXT"
    NAME="countdown_1" SIZE="10" VALUE=" "></P>

    <P><STRONG>Quante sono le dita di una mano?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">0 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">1 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">2 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">3 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">4 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">5 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">6 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">7 dita</LI>
    </OL>

    <P>Tempo a disposizione: <INPUT TYPE="TEXT"
    NAME="countdown_2" SIZE="10" VALUE=" "></P>

    <P><STRONG>Quante sono le dita di un piede?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">0 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">1 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">2 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">3 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">4 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">5 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">6 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_un_piede">7 dita</LI>
    </OL>

    <P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>

</FORM>

</BODY>
</HTML>
```

54.8.1 Utilizzo del programma

Il file mostrato è una pagina HTML che incorpora il codice JavaScript; per visualizzarla occorre un navigatore comune, in grado di eseguire il codice JavaScript. Potrebbe apparire come nella figura successiva:

Come si vede, appare un modello da compilare; al termine si deve selezionare il bottone `CONCLUDI LA VERIFICA` per ottenere l'esito. Ecco cosa si potrebbe ottenere se sono state date tutte le risposte in modo esatto, ma con un leggero ritardo rispetto ai tempi concessi:

In condizioni normali, dovrebbe apparire anche una finestra di conferma per la stampa dell'esito della verifica.

Si può osservare che la pagina normale della verifica viene azzerata completamente a ogni tentativo di ripristino della pagina e di azzeramento del tempo, in modo da scoraggiare un uso scorretto della verifica.

54.8.2 Codice HTML

Il codice HTML significativo è quello contenuto nell'elemento '**`BODY`**'. La prima cosa che si può osservare è l'associazione dell'evento '**`onLoad`**' alla chiamata della funzione ***document.test_form_1.reset()*** e successivamente della funzione ***countdown()***:

```
<BODY
  onLoad="document.test_form_1.reset(); countdown (1 * 60000);">
```

In pratica, la prima delle due funzioni esiste in quanto è stato definito il modulo (elemento '**`FORM`**') denominato '**`test_form_1`**' e serve ad azzerarne il contenuto; la seconda funzione è dichiarata nel codice JavaScript che appare nell'intestazione della pagina e il suo scopo è quello di azzerare il conto alla rovescia, in modo che parta da 60 000 ms (pari a 60 s, ovvero un minuto).

Con l'evento '**`onLoad`**' associato in questo modo alle due funzioni appena descritte, si vuole fare in modo che a ogni caricamento della pagina il suo contenuto si azzeri e il conteggio del tempo a disposizione riparta. Questa accortezza serve a evitare che si possa barare sulla durata, lasciando comunque la possibilità di ricominciare, prima di avere concluso la verifica, purché ci sia il tempo complessivo necessario.

Successivamente, dopo un titolo, si vede un elemento '**FORM**', che contiene in pratica le varie componenti da compilare della verifica: alcuni campi descrittivi, due gruppi di bottoni di selezione e il bottone di conclusione della verifica:

```
<FORM NAME="test_form_1" METHOD="get" ACTION="">
    ...
    <P>Cognome, nome, classe:
    <INPUT TYPE="TEXT" SIZE="60" NAME="student" VALUE="">
    </P>
    ...
    <P><STRONG>Quante sono le dita di una mano?</STRONG></P>
    <OL>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">0 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">1 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">2 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">3 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">4 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">5 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">6 dita</LI>
    <LI><INPUT TYPE="RADIO"
               NAME="dita_di_una_mano">7 dita</LI>
    </OL>
    ...
    <P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>
</FORM>
```

L'elenco numerato ('**OL**') contenente gli elementi di selezione di tipo '**RADIO**' (che pertanto consentono una sola selezione nel gruppo), serve a definire la scelta di una domanda a risposta multipla.

Al bottone finale viene abbinato l'evento '**onClick**', in modo da avviare contestualmente la funzione ***check_test_form_1()***, fornendo alla stessa l'orario esatto di inizio della verifica.

Si può osservare che inizialmente, l'elemento '**FORM**' contiene anche l'attributo '**METHOD**' e '**ACTION**', che però sono perfettamente inutili, dal momento che non si usa alcun programma CGI.

Qua e là, appare anche un campo particolare, che ha lo scopo di ospitare l'indicazione del tempo a disposizione, che decresce progressivamente:

```
<P>Tempo a disposizione: <INPUT TYPE="TEXT"
NAME="countdown_2" SIZE="10" VALUE=" "></P>
```

In questo campo non si deve scrivere, ma anche se fosse, il contenuto verrebbe rimpiazzato periodicamente con l'informazione del tempo che decresce.

54.8.3 Variabili globali

All'inizio del programma appare la dichiarazione di una variabile globale:

```
<SCRIPT LANGUAGE="JavaScript" TYPE="text/javascript">
<!--
    //
    // Get the initial time stamp.
    //
    var start_time = new Date ();
```

Lo scopo è quello di accumulare al caricamento della pagina la data e l'orario iniziale. Successivamente, questa informazione viene passata come argomento della funzione ***check_test_form_1()***, come già descritto in precedenza.

54.8.4 Conto alla rovescia

Una porzione del programma JavaScript è dedicata alla gestione di un orologio che mostra il tempo a disposizione all'interno dei campi del tipo:

```
<INPUT TYPE="TEXT" NAME="countdown_2" SIZE="10" VALUE=" ">
```


Questo lavoro è svolto dalla funzione ***countdown()***, che richiede come argomento il tempo a disposizione, espresso in millesimi di secondo:

```
function countdown (time_allowed)
{
    var hours;
    var minutes;
    var seconds;
    var time;
    var time_string;
    var time_sign;
    //
    // Verify the countdown time sign.
    //
    if (time_allowed > 0)
      {
        time_sign = "";
      }
    else
      {
        time_sign = "-";
      }
    //
    // Find the absolute countdown value.
    //
    time = Math.abs (time_allowed);
    //
    hours   = (time - (time % 3600000)) / 3600000;
    time    = time - (hours * 3600000);
    minutes = (time - (time % 60000)) / 60000;
    time    = time - (minutes * 60000);
    seconds = (time - (time % 1000)) / 1000;
    //
    if (minutes < 10)
      {
        minutes = "0" + minutes;
      }
    if (seconds < 10)
      {
        seconds = "0" + seconds;
      }
    if (hours < 10)
      {
        hours = "0" + hours;
      }
    //
    time_string = time_sign + hours + ":"
                            + minutes + ":" + seconds;
    //
    // Show the time string inside some text field.
    //
    document.test_form_1.countdown_0.value = time_string;
    document.test_form_1.countdown_1.value = time_string;
    document.test_form_1.countdown_2.value = time_string;
    //
    // Reduce 10 s inside the countdown variable.
    //
    time_allowed = time_allowed - 10000;
    //
    // Wait for 10 s for the recursive call.
    //
    setTimeout ("countdown (" + time_allowed + ")", 10000);
}
```

Il lavoro iniziale della funzione è quello di separare le varie componenti del tempo, in modo da individuare le ore, i minuti e i secondi, avendo anche l'accortezza di aggiungere uno zero iniziale quando il valore corrispondente ha una sola cifra. Il risultato di questa scomposizione viene aggregato nella variabile '**time_string**', in modo da produrre qualcosa di simile a «00:10:20» (zero ore, 10 minuti e 20 secondi), avendo cura anche di mostrare un segno se il valore è negativo; fatto questo, il valore della stringa viene copiato nei campi che servono a visualizzare il conteggio ('**document.test_form_1.countdown_*n*.value**').

Una volta provveduto a mostrare il valore aggiornato nei campi rispettivi, si decrementa il tempo a disposizione di 10 s e si esegue una chiamata ricorsiva attraverso l'aiuto della funzione ***setTimeout()***.

La funzione ***setTimeout()*** serve a eseguire una funzione, passata come stringa, dopo che è trascorso un certo tempo, senza bloccare il funzionamento del programma complessivo. In questo caso, si avvia la stessa funzione ***countdown()***, passando il valore aggiornato del tempo a disposizione, dopo che sono passati 10 s. Secondo la modalità normale di programmazione, sarebbe più logico eseguire un ciclo iterativo, inserendo una pausa in quel punto, ma in questo caso ciò non è possibile, sia perché manca una funzione appropriata nel linguaggio JavaScript, sia perché una tale funzione bloccherebbe il funzionamento del programma, impedendo così la compilazione del modulo e l'invio della verifica.

Una cosa da osservare è che il ciclo di aggiornamento dell'orologio che mostra il conto alla rovescia è di 10 s, mentre sarebbe facilissimo ridurlo a un solo secondo:

```
    //
    // Reduce 1 s inside the countdown variable.
    //
    time_allowed = time_allowed - 1000;
    //
    // Wait for a second for the recursive call.
    //
    setTimeout ("countdown (" + time_allowed + ")", 1000);
```

La scelta di avere un aggiornamento meno frequente dipende solo dall'intenzione di non agitare inutilmente chi sta svolgendo la verifica.

54.8.5 Scansione di un elenco di tipo «RADIO»

Le domande a risposta multipla vengono gestite attraverso componenti del modulo di tipo '**`RADIO`**'. L'insieme dell'elenco si ottiene tramite un nome del tipo seguente; in pratica, nel caso del primo elenco si tratta del nome '**`document.test_form_1.dita_di_una_mano`**':

```
document.form.radio
```

Questo nome rappresenta precisamente un array di valori logici, con cui si può poi verificare se il bottone corrispondente è stato premuto o meno:

```
if (document.form.radio[n].checked)
  {
    …
  }
```

Il primo elemento dell'array si raggiunge con l'indice zero e corrisponde al primo dei bottoni.

Nel programma JavaScript proposto viene usata una funzione apposita per la scansione di un gruppo di bottoni di questo tipo:

```
function scan_radio_button (radio_button)
{
    var counter;
    for (counter = 0;
         counter < radio_button.length;
         counter++)
      {
        if (radio_button[counter].checked)
          {
            return (counter + 1);
          }
      }
    return 0;
}
```

In pratica, attraverso la variabile '**`radio_button`**' si ottiene l'array da scandire, quindi si utilizza un contatore per verificare i vari elementi. Se si incontra un bottone premuto, la funzione termina il suo ciclo e restituisce il valore dell'indice aumentato di una unità, mentre se non viene trovato premuto alcun bottone, viene restituito il valore zero.

Come si comprende, la chiamata di questa funzione avviene in un modo simile a quello seguente:


```
dita_di_una_mano = scan_radio_button
                   (document.test_form_1.dita_di_una_mano);
```

54.8.6 Valutazione della verifica

Si ottiene la valutazione della verifica con chiamata della funzione ***check_test_form_1()*** che richiede di fornire la data e l'ora di inizio. La chiamata di questa funzione viene innescata dal bottone che si trova alla fine del modulo:

```
<P><INPUT TYPE="button" VALUE="concludi la verifica"
        onClick="check_test_form_1(start_time)"></P>
```

La funzione in questione contiene delle informazioni fisse, che potrebbero essere passate tramite gli argomenti. Si tratta precisamente del tempo a disposizione (che deve combaciare con quello stabilito per la chiamata della funzione che esegue il conto alla rovescia) e della penalità nel punteggio da applicare a ogni minuto di ritardo:

```
function check_test_form_1 (start_time)
{
    //
    var end_time = new Date ();
    var test_time = end_time - start_time;
    var time_allowed = 1 * 60000;
    var time_delay = test_time - time_allowed;
    //
    if (time_delay < 0)
      {
        time_delay = 0;
      }
    //
    // Penalty for a minute delay.
    //
    var score_time_penalty = 0.5;
```

Oltre a questo, c'è da osservare che la funzione contiene internamente l'informazione su quali sono le risposte corrette, attraverso degli array che contengono il punteggio da dare per ogni risposta. Si osservi che in questo caso, l'elemento iniziale (con indice zero) corrisponde alla mancanza di alcuna selezione:

```
    //
    // Set the result for every choice.
    //
    var score_dita_di_una_mano = new Array();
    score_dita_di_una_mano[0] = 0;
    score_dita_di_una_mano[1] = 0;
    score_dita_di_una_mano[2] = 0;
    score_dita_di_una_mano[3] = 0;
    score_dita_di_una_mano[4] = 0;
    score_dita_di_una_mano[5] = 0;
    score_dita_di_una_mano[6] = 5;
    score_dita_di_una_mano[7] = 0;
    score_dita_di_una_mano[8] = 0;
    //
    // Set the result for every choice.
    //
    var score_dita_di_un_piede = new Array();
    score_dita_di_un_piede[0] = 0;
    score_dita_di_un_piede[1] = 0;
    score_dita_di_un_piede[2] = 0;
    score_dita_di_un_piede[3] = 0;
    score_dita_di_un_piede[4] = 0;
    score_dita_di_un_piede[5] = 0;
    score_dita_di_un_piede[6] = 5;
    score_dita_di_un_piede[7] = 0;
    score_dita_di_un_piede[8] = 0;
```

Successivamente, nella funzione vengono salvati i dati descrittivi, che servono a identificare chi ha eseguito la verifica:

```
    //
    // Save the personal data.
    //
    var student = document.test_form_1.student.value;
    var date = document.test_form_1.date.value;
```

Si passa quindi alla scansione delle risposte e al calcolo del punteggio corrispondente:

```
    //
    // Scan the radio buttons.
    //
    var dita_di_una_mano
            = scan_radio_button
                    (document.test_form_1.dita_di_una_mano);
    //
    var dita_di_un_piede
            = scan_radio_button
                    (document.test_form_1.dita_di_un_piede);
    //
    // Calculate the results.
    //
    var account_score_time_penalty;
    var account_score_dita_di_una_mano;
    var account_score_dita_di_un_piede;
    var account_score_sum = 0;
    //
    account_score_time_penalty
            = (time_delay / 1000)
               * (score_time_penalty / 60);
    //
    account_score_dita_di_una_mano
        = score_dita_di_una_mano[dita_di_una_mano];
    account_score_dita_di_un_piede
        = score_dita_di_un_piede[dita_di_un_piede];
    account_score_sum
        = account_score_sum - account_score_time_penalty;
    account_score_sum
        = account_score_sum
          + account_score_dita_di_una_mano;
    account_score_sum
        = account_score_sum
          + account_score_dita_di_un_piede;
```

Terminata l'acquisizione e valutazione dei dati, si ha cura di azzerare il modulo, in modo da assicurare che non possa essere utilizzabile il bottone per il ritorno alla pagina precedente tramite il navigatore ipertestuale per modificare la verifica già terminata:

```
    //
    // Reset the form to avoid the use of the [Back] button
    // from the browser.
    //
    document.test_form_1.reset();
```

Successivamente si passa alla fase di produzione di una nuova pagina HTML con l'esito della verifica. Si osservi che a seconda dell'interprete JavaScript usato, l'anno potrebbe essere visualizzato correttamente o meno (per esempio, l'anno 2005 potrebbe essere mostrato come 105), ma si tratta di un errore trascurabile per i fini di questo prototipo:

```
    //
    // Print a simple HTML header.
    //
    document.writeln ("<HTML>");
    document.writeln ("<HEAD>");
    document.write   ("<TITLE>");
    document.write   ("Risultato della verifica");
    document.writeln ("</TITLE>");
    document.writeln ("</HEAD>");
    document.writeln ("<BODY>");
    //
    // Print a H1 title and some personal data.
    //
    document.writeln ("<H1>Verifica banale</H1>");
    document.writeln ("<P>studente: " + student + "</P>");
    document.writeln ("<P>data: " + date + "</P>");
    //
    // Print about time.
    //
    document.write ("<P>inizio della verifica: ");
    document.write (start_time.getYear());
    document.write (".");
    document.write (start_time.getMonth() + 1);
    document.write (".");
    document.write (start_time.getDate());
    document.write (" ");
    document.write (start_time.getHours());
```



```
    document.write (":");
    document.write (start_time.getMinutes());
    document.write (".");
    document.write (start_time.getSeconds());
    document.writeln ("</P>");
    //
    document.write ("<P>conclusione della verifica: ");
    document.write (end_time.getYear());
    document.write (".");
    document.write (end_time.getMonth() + 1);
    document.write (".");
    document.write (end_time.getDate());
    document.write (" ");
    document.write (end_time.getHours());
    document.write (":");
    document.write (end_time.getMinutes());
    document.write (".");
    document.write (end_time.getSeconds());
    document.writeln ("</P>");
    //
    document.write ("<P>durata complessiva della ");
    document.write ("verifica: ");
    document.write (test_time / 1000);
    document.write (" s, ");
    document.write (test_time / 1000 / 60);
    document.write (" m;");
    document.writeln ("</P>");
    //
    document.write ("<P>tempo a disposizione: ");
    document.write (time_allowed / 1000);
    document.write (" s; ");
    document.write ("ritardo: ");
    document.write (time_delay / 1000);
    document.write (" s; ");
    document.write ("penalit&agrave;/minuto: ");
    document.write (score_time_penalty);
    document.write ("; ");
    document.write ("penalit&agrave; complessiva: ");
    document.write (account_score_time_penalty);
    document.writeln ("</P>");
    //
    // First question.
    //
    document.write ("<P>Quante sono le dita di una mano? ");
    document.write ("risposta n. ");
    document.write (dita_di_una_mano);
    document.write ("; ");
    document.write ("punteggio: ");
    document.write (account_score_dita_di_una_mano);
    document.writeln ("</P>");
    //
    // Second question.
    //
    document.write ("<P>Quante sono le dita di un piede? ");
    document.write ("risposta n. ");
    document.write (dita_di_un_piede);
    document.write ("; ");
    document.write ("punteggio: ");
    document.write (account_score_dita_di_un_piede);
    document.writeln ("</P>");
    //
    // Final result.
    //
    document.write ("<P>Punteggio complessivo della ");
    document.write ("verifica: ");
    document.write (account_score_sum);
    document.writeln ("</P>");
    //
    // End of HTML.
    //
    document.writeln ("</BODY>");
    document.writeln ("</HTML>");
```

Al termine, si può osservare l'uso della funzione ***window.print()*** per invitare alla stampa del risultato, in modo da poterlo consegnare all'insegnante. Si osservi che con alcuni navigatori potrebbe anche non funzionare, ma rimane la possibilità di comandare la richiesta di stampa tramite il menù del navigatore stesso.

Alla fine della funzione, per completezza, viene usata la funzione

window.stop(), per interrompere il caricamento della pagina, che con alcuni navigatori rimane attivo, nonostante non ci sia altro da attendere.

```
    //
    // Print with a printer.
    //
    window.print ();
    //
    // Stop loading.
    //
    window.stop ();
}
```

54.9 HTML2ps

HTML2ps[5] è un programma in grado di comporre uno o più file HTML, generando un risultato in PostScript. Questo si ottiene attraverso l'aiuto di altri programmi che devono essere installati, come per esempio TeX.

Teoricamente, HTML2ps è in grado di ricomporre assieme un documento suddiviso su più file HTML, ma questa possibilità dipende molto dall'organizzazione di questi file, all'interno dei quali, i riferimenti ipertestuali devono essere molto semplici. In generale, è possibile l'acquisizione diretta dalla rete; tuttavia, sarebbe consigliabile prima la riproduzione locale, con l'ausilio di Wget (40.8), attraverso il quale si possono modificare automaticamente i riferimenti ipertestuali, rendendo omogeneo il tutto.

HTML2ps si compone semplicemente dell'eseguibile '**`html2ps`**' (un programma scritto in Perl) e di uno o più file di configurazione. È indispensabile almeno il file di configurazione generale, '`/etc/html2psrc`', che dovrebbe essere già predisposto in modo sufficientemente buono dal sistema di installazione. Eventualmente, gli utenti possono preparare una configurazione personalizzata nel file '`~/.html2psrc`' e altri file specifici da richiamare con l'opzione '**`-f`**', oltre all'aggiunta di stili ulteriori (opzione '**`-s`**').

54.9.1 Configurazione di HTML2ps

Come accennato, la configurazione di HTML2ps è indispensabile. Di solito si predispone almeno il file di configurazione generale, '`/etc/html2psrc`', mentre gli utenti hanno la possibilità di modificare o aggiungere qualcosa attraverso il file '`~/.html2psrc`'. La sintassi per la scrittura di questi file è la stessa dei fogli di stile CSS (sezione 54.6), con l'aggiunta di un selettore specifico, '**`@html2ps`**', che serve a indicare gli aspetti particolari che riguardano HTML2ps e non possono appartenere ai fogli di stile CSS.

Bisogna tenere presente che HTML2ps è in grado di riconoscere solo una parte limitata delle dichiarazioni CSS.

HTML2ps riconosce anche i commenti CSS e le inclusioni di file di configurazione aggiuntivi, secondo la forma:

```
@include file
```

Per cominciare, è opportuno vedere un esempio abbastanza semplice di ciò che potrebbe contenere un file di configurazione, quando questo viene generato automaticamente dalla procedura di installazione.

```
/* Configurazione globale per html2ps */

@html2ps {
  package {
    ImageMagick: 1;
    PerlMagick: 1;
    TeX: 1;
    Ghostscript: 1;
    check: weblint;
    libwww-perl: 1;
    path: "/usr/bin/X11:/usr/bin";
  }
  paper {
    type: A4;
  }
  option {
    hyphenate: 0;
  }
}
```


Si può osservare che in questo esempio è stata dichiarata solo la regola corrispondente al selettore '**`@html2ps`**', all'interno della quale si trovano altre sottoregole. Generalmente, le regole tipiche di uno stile CSS si aggiungono sotto. La configurazione predefinita dello stile CSS è indicata nella pagina di manuale *html2psrc(5)* e da questa si intende quali siano le possibilità effettive di HTML2ps nel riconoscere le dichiarazioni CSS:

```
BODY {
    font-family: Times;
    font-size: 11pt;
    text-align: left;
    background: white;
}

H1, H2, H3, H4, H5, H6 {
    font-weight: bold;
    margin-top: 0.8em;
    margin-bottom: 0.5em;
}

H1 { font-size: 19pt }
H2 { font-size: 17pt }
H3 { font-size: 15pt }
H4 { font-size: 13pt }
H5 { font-size: 12pt }
H6 { font-size: 11pt }

P, OL, UL, DL, BLOCKQUOTE, PRE {
    margin-top: 1em;
    margin-bottom: 1em;
}

P {
    line-height: 1.2em;
    text-indent: 0;
}

OL, UL, DD { margin-left: 2em }

TT, KBD, PRE { font-family: Courier }

PRE { font-size: 9pt }

BLOCKQUOTE {
    margin-left: 1em;
    margin-right: 1em;
}

ADDRESS {
    margin-top: 0.5em;
    margin-bottom: 0.5em;
}

TABLE {
    margin-top: 1.3em;
    margin-bottom: 1em;
}

DEL { text-decoration: line-through }

A:link, HR { color: black }
```

54.9.1.1 Configurazione della regola corrispondente al selettore speciale @html2ps

La regola corrispondente al selettore '**`@html2ps`**' si compone di dichiarazioni e di altre sottoregole per la configurazione di HTML2ps. Nelle sezioni seguenti vengono descritti i selettori specifici di queste sottoregole.

Alcune proprietà hanno un significato booleano. A loro si assegna il valore zero per indicare *Falso* e il valore uno per indicare *Vero*.

I valori che fanno riferimento a un'unità di misura, vanno indicati come avviene nei fogli di stile CSS: il numero seguito immediatamente dall'unità di misura. La tabella 54.172 elenca le unità di misura e le sigle corrispondenti che si possono utilizzare in questa circostanza. È importante osservare che l'unica dimensione relativa riconosciuta da HTML2ps è il quadratone e non sono previste misure percentuali come invece si può fare secondo le specifiche di W3C per i fogli di stile CSS.

Tabella 54.172. Unità di misura secondo HTML2ps.

Sigla	Unità di misura
cm	Centimetri.
mm	Millimetri.
pt	Punti tipografici.
pc	Pica.
em	Quadratoni, corrispondente alla dimensione della lettera «M» maiuscola.

Tabella 54.173. Proprietà del selettore '**@html2ps**'.

Proprietà	Descrizione
`numberstyle: 0|1`	Permette di stabilire la numerazione delle pagine: zero richiede l'uso dei numeri arabi; uno corrisponde a numeri romani. Il valore predefinito per questa proprietà è il valore zero.
`showurl: 0|1`	Attivando questa proprietà booleana, si ottiene l'inserimento nella composizione dell'indirizzo URI corrispondente ai riferimenti ipertestuali. In situazioni normali questo non avviene.
`seq-number: 0|1`	Permette di abilitare la numerazione dei titoli '**H1**', '**H2**',... '**H6**'. In condizioni normali, questo non avviene.

La sottoregola '**package**' serve a definire la disponibilità o meno di altri programmi di cui HTML2ps potrebbe avere bisogno. Di conseguenza si tratta di assegnamenti di valori booleani, dove zero rappresenta l'assenza del programma in questione e in generale è anche il valore predefinito.

```
@html2ps {
  …
  package {
    proprietà
    …
  }
  …
}
```

Tabella 54.174. Proprietà della sottoregola '**package**'.

Proprietà	Descrizione
`PerlMagick: 0|1`	Indica la mancanza o la disponibilità di PerlMagick.
`ImageMagick: 0|1`	Indica la mancanza o la disponibilità di ImageMagick.
`Ghostscript: 0|1`	Indica la mancanza o la disponibilità di Ghostscript.
`TeX: 0|1`	Indica la mancanza o la disponibilità di TeX.
`dvips: 0|1`	Indica la mancanza o la disponibilità di '**dvips**'.
`libwww-perl: 0|1`	Indica la mancanza o la disponibilità del modulo Perl Libwww-Perl.
`path:` ***percorsi_aggiuntivi***	Si tratta dell'indicazione di percorsi aggiuntivi per la ricerca degli eseguibili. Serve a garantire che i programmi utilizzati da HTML2ps siano raggiungibili per tutti gli utenti. In generale, in presenza di un sistema configurato bene, non dovrebbe essere necessaria l'indicazione di questa dichiarazione.

La sottoregola '**paper**' serve a definire le caratteristiche della carta. In generale si tratta solo delle dimensioni.

```
@html2ps {
  …
  paper {
    proprietà
    …
  }
  …
}
```

Tabella 54.175. Proprietà della sottoregola '**paper**'.

Proprietà	Descrizione
`type:` ***tipo_di_carta***	La direttiva serve a definire le dimensioni della carta, attraverso l'indicazione di un nome standard; per esempio: '**A0**', '**A1**', ... '**A10**', '**B0**', '**B1**', ... '**B10**', '**letter**', '**legal**', ecc. In alternativa, si possono indicare le dimensioni precise attraverso le proprietà '**height**' e '**width**'.
`height:` ***dimensione_assoluta***	Permette di definire l'altezza del foglio.
`width:` ***dimensione_assoluta***	Permette di definire la larghezza del foglio.

La sottoregola '**option**' serve a definire l'utilizzo di alcune opzioni, a cui si può accedere anche attraverso la riga di comando. Vengono descritte prima le dichiarazioni da indicare nel file di configurazione e poi le opzioni corrispondenti della riga di comando.

```
@html2ps {
  …
  option {
    proprietà
    …
  }
  …
}
```

Tabella 54.176. Proprietà della sottoregola '**option**' e opzioni corrispondenti della riga di comando.

Proprietà e opzione della riga di comando	Descrizione

`twoup: 0|1`
`-2`
`--twoup`
	Se attivato, fa in modo di ottenere un testo organizzato su due colonne verticali.

`toc: {f|h|t}[b]` `-C {f|h|t}[b]`	Fa in modo che venga generato un indice generale, in base alle opzioni specificate da una o più lettere: la lettera '**b**' richiede che l'indice generale sia collocato all'inizio; la lettera '**f**' richiede che l'indice generale sia generato a partire dai riferimenti contenuti nel documento; la lettera '**h**' richiede che l'indice generale sia generato a partire dai titoli definiti dagli elementi HTML da '**H1**' a '**H6**'; la lettera '**t**' richiede che l'indice generale sia generato a partire da elementi '**LINK**' contenenti l'attributo '**REV=TOC**'.

Proprietà e opzione della riga di comando	Descrizione

`DSC: 0|1`
`-D`
`--DSC`
	Se attivato, fa in modo di generare un file PostScript aderente alle specifiche DSC. In generale, per ottenere un file PostScript completo, è necessario attivare questa opzione.

`encoding:` ***codifica*** `-e` ***codifica*** `--encoding` ***codifica***	Permette di definire la codifica in cui è realizzato il file HTML. Il valore predefinito è '**`ISO-8859-1`**', ma sono poche altre le possibilità (si deve consultare la pagina di manuale).

`hyphenate: 0|1`
`-H`
`--hyphenate`
	Se attivato, fa in modo che il testo possa essere separato in sillabe, per facilitare l'impaginazione.

`language:` ***linguaggio*** `-l` ***linguaggio*** `--language` ***linguaggio***	Permette di indicare un linguaggio diverso da quello che può essere stato dichiarato nell'elemento '**`BODY`**' con l'attributo '**`LANG`**' di un documento HTML. La stringa che definisce il linguaggio va scelta in base a quanto già consentito dall'HTML (tabella 13.4).

`landscape: 0|1`
`-L`
`--landscape`
	e attivato, genera pagine orientate in modo orizzontale.

`number: 0|1`
`-n`
`--number`
	e attivato, fa in modo di aggiungere i numeri di pagina.

`startno:` ***n*** `-N` ***n*** `--startno` ***n***	Specifica il numero iniziale delle pagine. Il valore predefinito è uno.

`xref: 0|1`
`-R`
`--xref`
	Se attivato, fa in modo di aggiungere dei riferimenti visivi nel testo, in corrispondenza di quelli ipertestuali contenuti nel documento HTML.

`scaledoc:` ***scala_percentuale*** `-s` ***scala_percentuale*** `--scaledoc` ***scala_percentuale***	Riduce o amplia la scala del documento: il valore unitario rappresenta la situazione normale, di una scala pari al 100 %; valori superiori indicano un ingrandimento, mentre valori inferiori indicano una riduzione (si usa il punto per separare la parte intera dalle cifre decimali).

Proprietà e opzione della riga di comando	Descrizione

`web: {a|b|l|r|s}[p|L|`***n***`]`
`-W {a|b|l|r|s}[p|L|`***n***`]`
`-web {a|b|l|r|s}[p|L|`***n***`]`
	Fa in modo che vengano utilizzati più file HTML che si ritiene facciano parte dello stesso documento. Il modo in cui vengono presi in considerazione questi file dipende dalla stringa composta nel modo mostrato dallo schema sintattico.
	Si utilizza la lettera '**`a`**' per seguire tutti i riferimenti ipertestuali; la lettera '**`b`**' per seguire soltanto i riferimenti ipertestuali che riguardano la stessa directory del file iniziale; la lettera '**`l`**' per seguire soltanto i riferimenti ipertestuali che contengono l'attributo '**`REL=NEXT`**' all'interno dell'elemento '**`LINK`**'; la lettera '**`r`**' per seguire soltanto i riferimenti ipertestuali relativi; la lettera '**`s`**' per seguire solo i riferimenti allo stesso nodo del documento di partenza.
	Inoltre, la lettera '**`p`**' fa in modo che sia chiesta conferma per ogni file HTML da aggiungere (ciò avviene in ogni caso quando si superano i 50 file); la lettera '**`L`**' serve a riordinare i documenti in base alla struttura gerarchica; un numero intero ('***n***') un numero indica il livello massimo di ricorsione, tenendo conto che il valore predefinito è di quattro livelli.

La sottoregola '**`margin`**' permette di definire esplicitamente i margini della pagina.

```
@html2ps {
  …
  margin {
    proprietà
    …
  }
  …
}
```

Questa sottoregola è superata e viene sostituita dalla configurazione nel file di stile CSS, utilizzando la regola '**`@page`**', introdotta dalle specifiche CSS2.

Tabella 54.177. Proprietà della sottoregola superata '**`margin`**'.

Proprietà	Descrizione
`left` ***margine_sinistro*** `right` ***margine_destro***	Indicano i margini sinistro e destro rispettivamente. Il valore predefinito è '**`2.5cm`**', pari a 2,5 cm.
`top` ***margine_superiore*** `bottom` ***margine_inferiore***	Indicano i margini superiore e inferiore rispettivamente. Il valore predefinito è '**`3cm`**', pari a 3 cm.
`middle` ***distanza_tra_colonne***	Indica la distanza orizzontale tra le colonne, quando si stampano due colonne per pagina. Il valore predefinito è '**`2cm`**', pari a 2 cm.

La sottoregola '**`xref`**' permette di definire esplicitamente il modo in cui vengono indicati i riferimenti nel testo, quando questa funzionalità è stata abilitata.

```
@html2ps {
  …
  xref {
    proprietà
    …
  }
  …
}
```

Tabella 54.178. Proprietà della sottoregola '**xref**'.

Proprietà	Descrizione
`text:` ***modello***	Permette di definire il modello da utilizzare, tenendo conto che il simbolo '**$N**' viene rimpiazzato con il numero della pagina. Il modello predefinito è '**[p $N]**'.
`passes:` ***n***	Permette di definire il numero di passaggi necessario per determinare in modo corretto i riferimenti incrociati. Il valore predefinito è il valore uno, ma l'inserzione del testo corrispondente al modello potrebbe cambiare la sequenza delle pagine, per cui si potrebbe rendere necessario un numero maggiore di passaggi.

La sottoregola '**quote**' permette di definire esplicitamente l'uso delle virgolette più appropriate in base al linguaggio. Queste virgolette vengono inserite nel testo in corrispondenza degli elementi '**Q**'. In generale, i valori predefiniti per la lingua italiana sono già corretti. Viene mostrato solo un esempio per comprendere intuitivamente come si potrebbe adoperare questa sottoregola:

```
quote {
  it {
    open: "«";
    close: "»";
    open2: "‘";
    close2: "’";
  }
}
```

Si intende dall'esempio che sono disponibili solo due livelli di virgolette.

La sottoregola '**toc**' permette di definire alcune caratteristiche relative all'indice generale, quando la sua realizzazione è stata richiesta espressamente. In particolare si può utilizzare la proprietà '**level**' alla quale si assegna un numero, che sta a indicare i livelli da prendere in considerazione. Il valore predefinito è sei, che produce una voce per ogni tipo di titolo '**Hn**' (da '**H1**' a '**H6**').

La sottoregola '**hyphenation**' permette di definire la collocazione del file TeX contenente i modelli per la separazione in sillabe. La cosa si fa distinguendo tra diversi linguaggi. L'esempio seguente dovrebbe essere sufficiente a intendere intuitivamente la cosa:

```
hyphenation {
  it {
    file: "/usr/share/texmf/tex/generic/hyphen/ithyph.tex";
  }
  en {
    file: "/usr/share/texmf/tex/generic/hyphen/ushyph1.tex";
  }
}
```

Le sottoregole '**header**' e '**footer**' permettono di definire l'intestazione e il fondo pagina, dove di solito si collocano alcune informazioni ricorrenti assieme al numero della pagina. Le proprietà di queste sottoregole sono praticamente le stesse; qui vengono elencate solo alcune di queste proprietà nella tabella 54.182. La tabella 54.181 elenca alcuni simboli che possono essere utilizzati per definire i modelli delle intestazioni e dei fondo pagina.

Tabella 54.181. Simboli utilizzabili nelle intestazioni e nei fondo pagina.

Simbolo	Corrispondenza
`$T`	Titolo del documento.
`$A`	Autore, come specificato in '**<META NAME="Author" CONTENT="…">**'.
`$U`	URI del documento.
`$N`	Numero di pagina.
`$H`	Titolo attuale ('**H1**'...'**H3**').
`$D`	Data e orario attuale.
`\$`	Dollaro.

Tabella 54.182. Alcune proprietà utilizzabili nell'intestazione e nel fondo delle pagine.

Proprietà	Contenuto
`left`	Intestazione allineata a sinistra.
`center`	Intestazione al centro.
`right`	Intestazione allineata a destra.
`odd-left`	Intestazione delle pagine dispari allineata a sinistra.
`odd-center`	Intestazione delle pagine dispari al centro.
`odd-right`	Intestazione delle pagine dispari allineata a destra.
`even-left`	Intestazione delle pagine pari allineata a sinistra.
`even-center`	Intestazione delle pagine pari al centro.
`even-right`	Intestazione delle pagine pari allineata a destra.
`font-family`	Tipo di carattere da usare (predefinito Helvetica).
`font-size`	Dimensione del carattere (predefinito 8 punti).
`font-style`	Forma del carattere (predefinita la forma normale).
`font-weight`	Spessore del carattere (predefinito lo spessore normale).

54.9.1.2 Configurazione in cascata

La configurazione di HTML2ps segue la logica dei fogli di stile CSS, anche per ciò che riguarda la sua definizione in cascata. In generale: il file '`/etc/html2psrc`' contiene le indicazioni essenziali; il file '`~/.html2psrc`' contiene la configurazione personalizzata; l'opzione '**-f**' consente di aggiungere altra configurazione specifica; l'opzione '**-S**' consente di aggiungere una stringa ulteriore allo stile.

Quando si utilizza l'opzione '**-f**', se si vuole evitare di eliminare la configurazione standard dei file '`/etc/html2psrc`' e '`~/.html2psrc`', si deve iniziare con i due punti ('**:**'), come si vede nell'esempio seguente:

```
$ html2ps -f :locale manuale.html > manuale.ps [Invio]
```

Si possono anche sommare assieme più configurazioni o stili CSS locali, come si vede nell'esempio seguente, dove si utilizzano i file '`locale`', '`A4`' e '`numerato`':

```
$ html2ps -f :locale:A4:numerato ↩
↪        manuale.html > manuale.ps [Invio]
```

L'opzione '**-S**' serve solo per aggiungere una regola al volo, indicandola direttamente nella riga di comando, come si vede nell'esempio seguente:

```
$ html2ps -f :locale -S "H1 { color: blue }" ↩
↪        manuale.html > manuale.ps [Invio]
```

54.9.2 Avvio di HTML2ps

HTML2ps si utilizza attraverso l'eseguibile '**html2ps**', con la sintassi seguente:

```
html2ps opzioni [file_html]
```

Il file da convertire può essere indicato nella riga di comando, dove in tal caso può trattarsi anche di un URI, oppure può essere fornito attraverso lo standard input.

Quasi tutte le opzioni di questo programma sono richiamabili anche tramite una proprietà corrispondente nella sottoregola '**option**', come è già stato descritto. Qui vengono riepilogate le opzioni più importanti nella tabella 54.183. In particolare, si può osservare che si può indicare il nome del file da generare attraverso l'opzione '**-o**', oppure '**--output**', altrimenti il risultato della conversione viene emesso attraverso lo standard output.

Tabella 54.183. Riepilogo delle opzioni più comuni.

Opzione	Descrizione
`-2` `--twoup`	Due colonne verticali.
`-D` `--DSC`	Genera un file PostScript DSC (standard).
`-e` `--encoding`	Stabilisce la codifica originale.
`-H` `--hyphenate`	Abilita la separazione in sillabe.
`-L` `--landscape`	Orientamento orizzontale.
`-n` `--number`	Aggiunge i numeri alle pagine.
`-o` `--output`	Specifica il file PostScript da generare.
`-R` `--xref`	Mostra gli URI dei riferimenti ipertestuali.
`-s` `--scaledoc`	Cambia la scala del documento.
`-W` `--web`	Definisce come gestire più file HTML assieme.
`-f` `--rcfile`	Specifica i file di configurazione aggiuntivi o alternativi.
`-S` `--style`	Specifica una regola aggiuntiva al volo.

Segue la descrizione di alcuni esempi.

- `$ html2ps -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`'.

- `$ html2ps -2 -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che risulta organizzato in due colonne verticali.

- `$ html2ps -R -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che contiene dei riferimenti incrociati visibili.

- `$ html2ps -2 -s 0.5 -o documento.ps documento.html` [*Invio*]

  Converte il file '`documento.html`' nel file '`documento.ps`', che risulta organizzato in due colonne verticali, con la dimensione del carattere ridotta alla metà.

- `$ html2ps -W b -o XFree86.ps ↩`
  `↪ XFree86-Video-Timings-HOWTO.html` [*Invio*]

  Converte i file HTML che iniziano da '`XFree86-Video-Timings-HOWTO.html`' in un solo file PostScript, denominato '`XFree86.ps`'. In particolare viene richiesto di seguire solo i riferimenti ipertestuali rivolti alla stessa directory di partenza.

54.9.3 Particolarità nell'HTML

HTML2ps interpreta alcuni «comandi» speciali all'interno del file HTML. Si tratta di:

- salto pagina incondizionato, il quale si ottiene con uno dei comandi seguenti:

```
<HR class=PAGE-BREAK>
```

```
<?page-break>
```

```
<!--NewPage-->
```

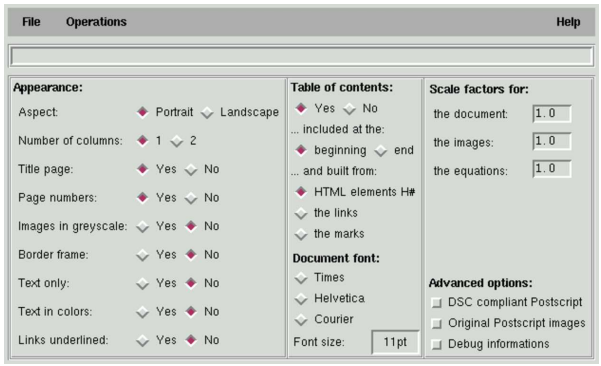
- esclusione di parte del testo dalla composizione stampata, attraverso un elemento '**DIV**' speciale:

```
<DIV class=NOPRINT>
    ...
    <!-- Testo che viene ignorato da HTML2ps -->
    ...
</DIV>
```

54.9.4 Programma frontale per semplificare l'utilizzo di HTML2ps

Assieme a HTML2ps si dovrebbe trovare un programma aggiuntivo che facilita il suo utilizzo attraverso un pannello grafico. Si tratta dell'eseguibile '**xhtml2ps**', che si vede in particolare nella figura 54.188. Il suo utilizzo dovrebbe essere intuitivo, dal momento che si rifà alle opzioni delle riga di comando.

Figura 54.188. Programma frontale per il controllo di HTML2ps.



54.10 Introduzione a Amaya

Amaya[6] è un sistema visuale integrato di navigazione e composizione di documenti HTML e XHTML. È interessante notare che Amaya è anche in grado di riconoscere e utilizzare i fogli di stile CSS.

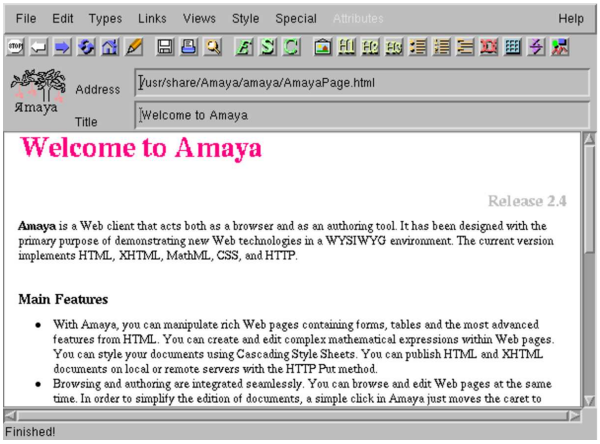
Amaya è disponibile per vari sistemi operativi. Probabilmente, questo fatto ha spinto gli sviluppatori del programma a costruire per lui un mondo a parte. In particolare, la tastiera viene gestita da Amaya in modo indipendente dal sistema sottostante.

L'avvio di Amaya è molto semplice, attraverso l'eseguibile '**amaya**', dal momento che gli argomenti sono tutti facoltativi:

```
amaya [-display schermo] [file | uri]
```

La figura 54.189 mostra come si presenta all'avvio, quando non si indica alcun file.

Figura 54.189. Amaya.



Amaya è un sistema di composizione HTML e XHTML, molto sofisticato e molto serio nel suo approccio a questi formati. Qui si introduce semplicemente il suo utilizzo, tenendo conto che la documentazione originale, accessibile anche dal menù *Help*, è buona.

54.10.1 Navigazione e composizione

Amaya è sia un navigatore HTTP, sia un sistema di composizione in HTML. Questo fatto ha delle implicazioni nel suo utilizzo che a prima vista possono sembrare un po' strane, benché siano assolutamente logiche. Per prima cosa è importante sapere che è possibile controllare la modalità di accesso al documento, attraverso la voce *Editor Mode* del menù *Edit*. Attivandola si abilita la modifica del documento; disattivandola si richiede espressamente di accedere in sola lettura.

Quando Amaya accede in sola lettura, si comporta come un navigatore normale; quando è consentita la modifica, il documento può essere alterato e salvato successivamente.

Quando si accede a un riferimento ipertestuale, come si fa di solito con i navigatori, il documento che si ottiene può occupare la stessa finestra di partenza, oppure può essere messo in un'altra. La scelta è abbastanza logica: se il documento di partenza non è stato alterato, si utilizza la stessa finestra iniziale.

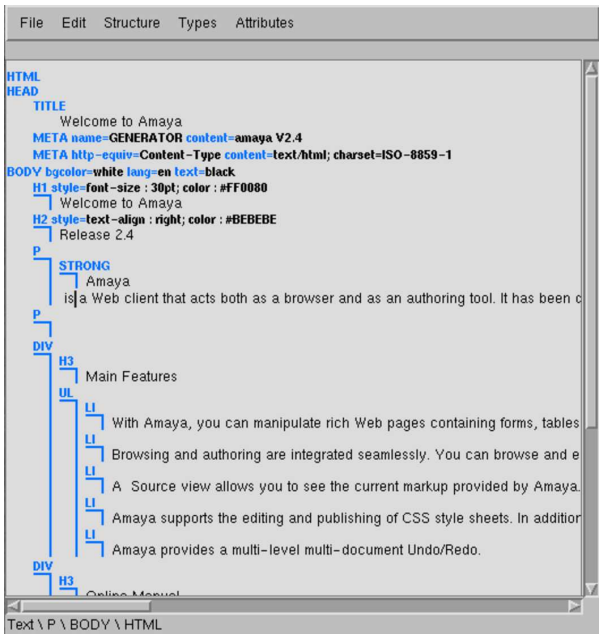
Per selezionare un riferimento ipertestuale, in condizioni normali serve un clic doppio con il primo tasto del mouse, perché con uno solo si posiziona semplicemente il cursore del testo. È possibile modificare la configurazione per fare in modo che basti un solo clic, ma in generale questa non è una buona idea, dal momento che diventerebbe difficile portare il cursore sopra un riferimento ipertestuale.

54.10.1.1 Modifica del documento

La modifica di un documento HTML può avvenire in modo visuale, diretto, attraverso la finestra che si usa anche per la sua lettura. La vera «forza» di Amaya sta nella possibilità di accedere al documento in una forma diversa, attraverso la sua struttura, in modo da avere una visione più chiara di ciò che si sta facendo.

Dal menù *Views* si possono selezionare le voci *Show structure* e *Show alternate*. La prima apre una finestra separata contenente la struttura, come si vede nell'esempio di figura 54.190, la seconda mostra il documento in un modo alternativo, precisamente in forma testuale senza grafica. La modifica in una di queste finestre si ripercuote simultaneamente su tutte le altre.

Figura 54.190. La visione della struttura.



Dallo stesso menù è possibile selezionare la voce *Show source* per accedere a una finestra contenente il sorgente del documento. Anche se è possibile modificare il testo direttamente nel sorgente, le modifiche non si applicano istantaneamente alle altre finestre, a meno di utilizzare la voce *Synchronize* dal menù *File*. Tuttavia, lo svantaggio nell'accedere direttamente al sorgente sta nel fatto che Amaya ha difficoltà a correggere gli errori nell'uso dell'HTML da parte di un autore inesperto, mentre nelle altre finestre questo non può avvenire, perché la struttura è sotto il pieno controllo del programma.

È interessante notare che alla base di ogni finestra utile per accedere alla modifica del documento appare l'indicazione sintetica della struttura del punto in cui si trova il cursore. Per esempio, la sequenza

```
Text \ P \ BODY \ HTML
```

indica che si tratta di testo contenuto in un elemento '**P**', che è contenuto nell'elemento '**BODY**', che a sua volta è parte dell'elemento '**HTML**':

```
<HTML>
    <BODY>
        <P>... <!-- Testo --> </P>
    </BODY>
</HTML>
```

Oltre alle specificità di Amaya, il suo funzionamento è abbastanza intuitivo. Si comprende che per poter essere utilizzato in modo conveniente, è più importante conoscere bene le potenzialità dell'HTML e dei fogli di stile CSS, prima di cercare di approfondire l'uso di questo programma.

54.10.2 Configurazione

La maggior parte della configurazione di Amaya è accessibile attraverso una delle voci del sottomenù *Preferences* del menù *Special*.

Nella directory personale dell'utente che utilizza il programma, Amaya crea la sottodirectory '`.amaya/`', in cui inserisce il file di configurazione generale '`thot.rc`', la sottodirectory per la sua memoria cache, '`libwww-cache/`' e i propri file temporanei. A parte la collocazione del file '`thot.rc`', il resto può essere spostato altrove attraverso la configurazione.

54.10.3 Aggregazione di un documento composto

Amaya è in grado di aggregare un documento composto da più «pagine» HTML in un solo file, attraverso la voce *Make book* del menù *Special*.

Per ottenere questo risultato si parte da un file HTML composto da un titolo contenuto in un elemento '**H1**', seguito da testo e da una serie di riferimenti. Questi riferimenti (l'elemento '**A**' con l'attributo '**HREF**') sono organizzati solitamente in un elenco puntato o numerato, ma a parte questo, tali riferimenti devono contenere anche l'attributo '**REL**', a cui viene assegnato il valore '**chapter**' o '**subdocument**'. L'esempio seguente rappresenta bene questa struttura di partenza:

```
<!DOCTYPE HTML PUBLIC
    "-//W3C//DTD HTML 4.0 Transitional//EN"
    "http://www.w3.org/TR/REC-html40/loose.dtd">
<html>
<head>
  <title>Using Amaya</title>
  <style type="text/css">
    BODY { background-color : #FFFFFF }
  </style>
</head>
<body lang="en">
<!-- ... -->
<h1 style="text-align : center">Using Amaya</h1>
<!-- ... -->
<p>Each following section gives a short description of how
to use a specific Amaya functionality.</p>
<ul>
  <li><a rel="Chapter"
  href="Browsing.html#Browsing">Browsing with Amaya</a></li>
  <li><a rel="Chapter"
  href="Selecting.html#Selecting">Selecting</a></li>
  <li><a rel="Chapter"
  href="Searching.html#Searching">Searching and replacing
  text</a></li>
  <li><a rel="Chapter" href="Views.html#Views">Displaying
  Views</a></li>
  <li><a rel="Chapter"
  href="Creating.html#Creating">Creating new
  elements</a></li>
<!-- ... -->
</ul>
<!-- ... -->
<p><a name="There">There is also a brief introduction </a>
which explains some of the different types that can be used
in Amaya such as headings, lists, and quotations, and how to
use them.</p>
<ul>
  <li><a href="HTML-elements/infoTypes.html"
  rel="Chapter">Information types in HTML</a></li>
  <li><a href="HTML-elements/structure.html"
  rel="Chapter">HTML Document Structure</a></li>
  <li><a href="HTML-elements/headings.html"
  rel="Chapter">Headers</a></li>
<!-- ... -->
</ul>
<hr>
<!-- ... -->
</body>
</html>
```

In particolare, l'elemento

```
<a href="HTML-elements/structure.html" rel="Chapter">HTML
Document Structure</a>
```

implica l'inclusione del corpo del file '`HTML-elements/structure.html`' in quel punto, al posto del suo riferimento.

Per la precisione, si possono distinguere questi casi: quando il riferimento è fatto a un documento completo, come appena visto, si ottiene l'inclusione del contenuto del suo elemento '**BODY**'; se invece il riferimento è fatto a un'etichetta di un certo elemento, viene incorporato solo il contenuto di quell'elemento.

Nella realizzazione di un documento articolato in più file differenti, converrebbe avere l'accortezza di delimitare la parte sostanziale del testo di ogni file HTML in un elemento '**DIV**' provvisto di etichetta a cui poter fare riferimento attraverso l'indice di partenza (l'attributo '**ID**'). In questo modo si potrebbero escludere dall'aggregazione una serie di informazioni che servono solo per la navigazione (pulsanti per avanzare, indietreggiare o raggiungere l'indice).

Un indice di partenza può anche fare riferimento a file che contengono a loro volta dei sottoindici, realizzando quindi una struttura ad albero abbastanza articolata. Amaya continua ad aggregare i file finché trova elementi '**A**' contenenti l'attributo '**REL**' a cui sono assegnate le parole chiave già indicate.

54.11 HTMLDOC

HTMLDOC[7] è un sistema di composizione basato su HTML. In pratica, si parte da uno o più file HTML e si ottiene una composizione in PostScript, PDF e HTML. I file HTML di partenza devono avere una struttura ragionevolmente semplice, dove il testo sia strutturato in capitoli, inizianti con un'intestazione '**H1**', suddivisi nel modo consueto, attraverso le intestazioni di livello inferiore ('**H2**', '**H3**',... '**H7**'). In questo modo, HTMLDOC è in grado di generare automaticamente un indice generale e diventa utile tutto il sistema anche quando l'obiettivo è la generazione di una composizione finale nello stesso formato HTML.

HTMLDOC è disponibile sia su piattaforme Unix, sia su sistemi MS-Windows. In particolare, può funzionare in modo interattivo, attraverso l'interfaccia grafica, oppure in modo non interattivo utilizzando semplicemente opzioni della riga di comando.

54.11.1 Sorgente HTML

Il sorgente di un documento HTMLDOC può essere contenuto in un file singolo, oppure in più file HTML distinti. La prima cosa che dovrebbe apparire nel corpo del file HTML è un'intestazione di tipo '**H1**'.

In questi file HTML non sono ammissibili le inserzioni di oggetti, a parte le immagini nel modo consueto, attraverso l'elemento '**IMG**'; non si possono usare le cornici (*frame*); vengono ignorati gli script e le applet. È possibile selezionare soltanto i tipi di carattere standard, corrispondenti a: Helvetica, Times e Courier. Dal momento che HTMLDOC è disponibile anche su piattaforma MS-Windows, il carattere Arial viene convertito automaticamente in Helvetica. Non sono gestiti i fogli di stile e le tabelle sono riconosciute solo al livello di HTML 3.2.

La documentazione di HTMLDOC è scritta in questo modo, per cui può essere osservata la tecnica utilizzata e il risultato che si ottiene. In generale, dopo l'elemento '**HEAD**', all'interno dell'elemento '**BODY**' si comincia subito con un'intestazione '**H1**'. L'esempio seguente è un estratto del file '`intro.html`', che è il primo del gruppo di file che compongono la documentazione di HTMLDOC:

```
<HTML>
<HEAD>
   <META HTTP-EQUIV="Content-Type"
         CONTENT="text/html; charset=iso-8859-1">
   <META NAME="author" CONTENT="Michael R. Sweet">
   <META NAME="copyright"
         CONTENT="Copyright 1997-1999, See the GNU General
                  Public License for Details.">
   <META NAME="Author"
         CONTENT="Michael R. Sweet, Easy Software Products">
   <TITLE>HTMLDOC 1.7 User's Guide</TITLE>
</HEAD>
<BODY>

<DIV ALIGN=right>
<H1>
Introduction</H1></DIV>

<H2>About This Software</H2>
This document describes how to use the <I>HTMLDOC</I>
software, version 1.7. <I>HTMLDOC</I> is a HTML document
processing program that generates indexed HTML, Adobe&reg;
PostScript<SUP>TM</SUP>, and Adobe Portable Document Format
```

```
(PDF 1.2) files suitable for printing or online viewing.

<P>No restrictions are placed upon the output produced by
<I>HTMLDOC</I>.

<H2>History</H2>

Like many programs <I>HTMLDOC</I> was developed in response
to a need my company had for generating high-quality
documentation in printed and
...
...

<H2>Why Just HTML?</H2>

Some people have asked why this program only deals with HTML
input files and is not able to read any Standard Generalized
Markup Language (SGML) file. The reasons are numerous but
basically boil down to:
...
...
</BODY>
</HTML>
```

Eventualmente è possibile convertire un file HTML singolo che non sia conforme a questa struttura, utilizzando un'opzione apposita, '**--webpage**', con la quale non si ottiene più l'indice generale, ma soprattutto non è possibile aggregare più file HTML assieme in un documento finale unico.

HTMLDOC è progettato per gestire documenti di dimensioni molto grandi; tuttavia esistono dei limiti, fissati nel file '`config.h`', che appartiene ai sorgenti del programma. Eventualmente si possono estendere tali limiti modificando questo file e ricompilando successivamente i sorgenti.

54.11.2 Funzionamento

HTMLDOC si compone dell'eseguibile '**htmldoc**', che può essere avviato senza argomenti, per ottenere un funzionamento interattivo:

```
htmldoc [opzioni file_html...]
```

È prevista la presenza di un file di configurazione personale, dei singoli utenti. Si tratta del file '`~/.htmldocrc`'. Questo file viene creato la prima volta dall'eseguibile '**htmldoc**' e potrebbe essere utile ritoccare la direttiva di dichiarazione del programma usato per modificare i file HTML sorgenti:

```
#HTMLDOCRC 1.7
EDITOR=amaya %s
```

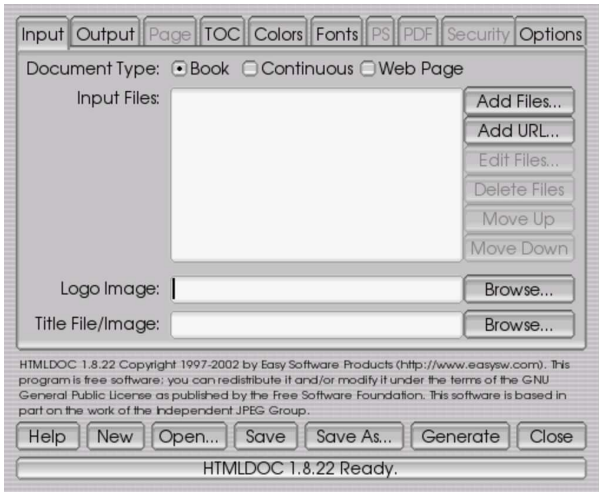
In questo caso, si fa in modo di utilizzare Amaya per la modifica del sorgente HTML, quando questo tipo di programma viene richiamato da HTMLDOC.

Purtroppo non è disponibile una documentazione sufficiente sulle direttive di questo file di configurazione.

Nel seguito viene mostrato il funzionamento interattivo di HTMLDOC, abbinando anche la descrizione delle opzioni che possono servire per ottenere lo stesso risultato senza interazione con il programma. Intanto, nella figura 54.197 si può osservare come appare inizialmente, dopo l'avvio con il comando seguente:

```
$ htmldoc [ Invio ]
```


Figura 54.197. Aspetto iniziale di HTMLDOC, quando viene avviato senza argomenti.



Nella parte bassa del pannellino grafico, appaiono alcuni pulsanti grafici, che fanno riferimento alla possibilità di creare e salvare un file contenente tutte le informazioni sulla composizione che si vuole generare. Questo file può essere inteso come la configurazione del libro che si vuole comporre.

- Il pulsante NEW serve a eliminare il lavoro in corso, per ricominciare con un altro nuovo.
- Il pulsante OPEN permettere di accedere al file system per selezionare un file contenente le informazioni su una composizione già predisposta in precedenza.
- I pulsanti SAVE e SAVE AS consentono di salvare l'impostazione attuale, eventualmente dando un nome nuovo alla cosa.
- Il pulsante GENERATE avvia la composizione in base alle informazioni indicate, generando uno o più file PostScript, PDF o HTML, a seconda di quanto specificato a questo proposito.
- Il pulsante CLOSE termina il funzionamento di HTMLDOC.

La parte centrale del pannello grafico di HTMLDOC cambia in funzione del lembo superiore selezionato.

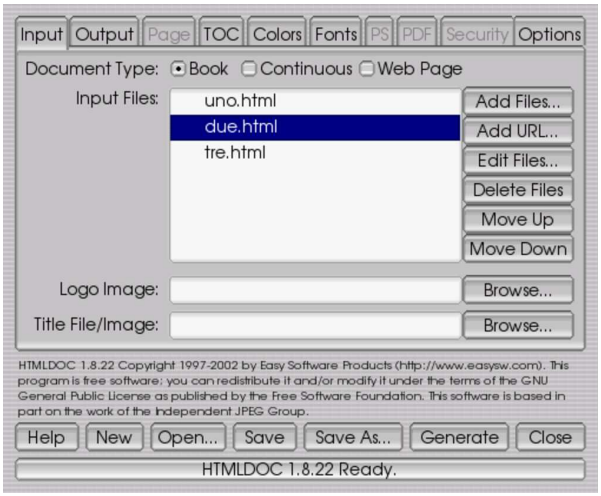
54.11.3 Definizione dei file sorgenti

La prima fase per arrivare alla composizione è quella della selezione dei file HTML che compongono i sorgenti. Ciò si fa dalla finestra che si ottiene selezionando l'etichetta *Input*, che corrisponde alla situazione in cui si presenta HTMLDOC all'avvio.

In alto, si può osservare la presenza di due pulsanti di selezione, dove si può specificare il tipo di sorgente che si utilizza. La voce BOOK indica l'intenzione di utilizzare uno o più file HTML per generare un documento unico, in forma di libro, mentre la voce WEB PAGE, corrispondente all'opzione '**--webpage**', specifica che si tratta di un solo file HTML che non ha la struttura richiesta per realizzare un libro.

Il pulsante grafico laterale ADD FILE consente di accedere al file system per selezionare i file HTML che compongono i sorgenti del documento che si vuole comporre. Nella figura 54.198 appaiono selezionati i file '`uno.html`', '`due.html`' e '`tre.html`'.

Figura 54.198. Sono stati selezionati tre file.



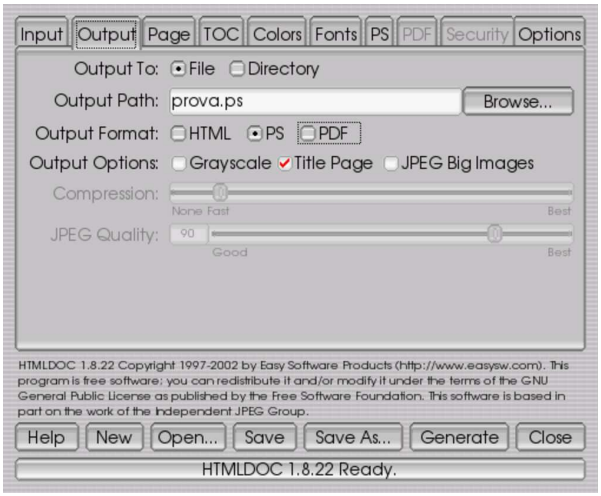
Sul lato destro si possono osservare anche altri pulsanti grafici, che si abilitano solo quando si seleziona uno o più nomi nell'elenco di quelli che compongono l'insieme dei sorgenti del documento: EDIT FILE permette di avviare automaticamente il programma per la modifica dei file HTML; DELETE FILE elimina i nomi evidenziati dall'elenco, ma senza cancellarli dal file system; MOVE UP e MOVE DOWN consentono di spostare il nome selezionato in alto o in basso, cambiando l'ordine.

Per completare l'estetica della composizione, è possibile specificare un'immagine da usare come logo e un'altra immagine da utilizzare nella copertina (ammesso che sia stata ablitata la sua generazione). Nel primo caso si può usare l'opzione '**`--logoimage`** ***file_immagine***'; nel secondo l'opzione '**`--titlefile`** ***file_immagine***'.

54.11.4 Composizione

L'etichetta *Output* consente di accedere alla definizione del file o dei file che si vogliono ottenere dalla composizione. Il risultato della composizione può essere un file oppure una directory, selezionando FILE oppure DIRECTORY, rispettivamente. Queste due voci corrispondono alle opzioni '**`--outfile`** ***file***' e '**`--outdir`** ***directory***'. Il file o la directory in questione si indicano sotto, mentre più giù si specifica esattamente il tipo di composizione che si vuole generare: HTML, PS, PS2 e PDF (opzione '**`--format {ps1|ps2|pdf|html}`**').

Figura 54.199. Definizione del risultato della composizione che si vuole ottenere.



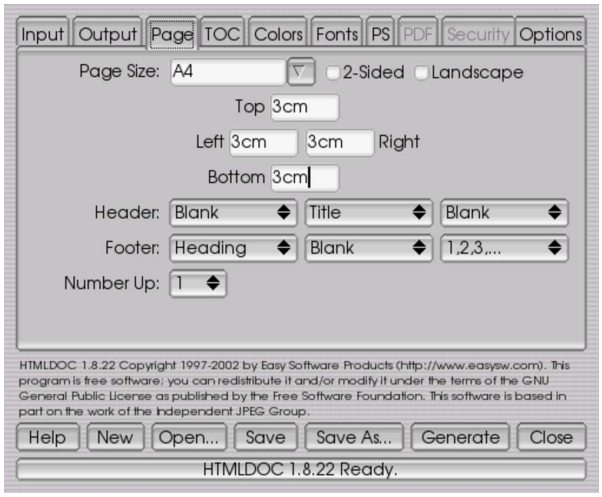
Il senso delle opzioni che appaiono nel resto della maschera è abbastanza intuitivo. Si può osservare il fatto che sia possibile cambiare il colore dello sfondo e anche utilizzare un'immagine per questo.

Se si indica una directory, si intende generare un gruppo di file nella directory stessa. Ciò può essere utile nella composizione in HTML, ma funziona nello stesso modo anche per le altre forme di composizione.

54.11.5 Formato e aspetto delle pagine

L'etichetta *Page* consente di accedere alla definizione delle pagine, nel caso in cui la composizione richiesta serva a generare un formato PostScript o PDF. È possibile indicare il formato della pagina (corrispondente all'opzione '**`--size`** ***formato***'), se la stampa avviene su entrambi i lati del foglio (la voce DOUBLE-SIDED, ovvero l'opzione '**`--duplex`**'), i margini (le opzioni '**`--left`** ***n*****`{in|cm|mm}`**', '**`--right`** ***n*****`{in|cm|mm}`**', '**`--top`** ***n*****`{in|cm|mm}`**', '**`--bottom`** ***n*****`{in|cm|mm}`**'), l'intestazione e il piè pagina (opzioni '**`--header`** ***xyz***' e '**`--footer`** ***xyz***').

Figura 54.200. Definizione della pagina nel caso di composizione per la stampa.

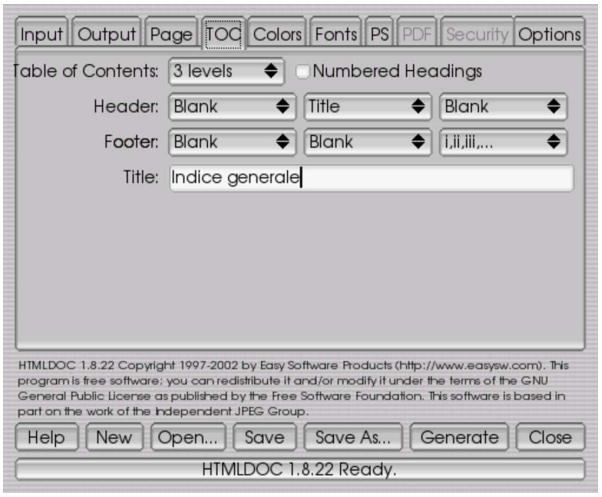


La selezione di intestazione e piè di pagina consente di definire ciò che deve apparire a sinistra, al centro e a destra di queste.

54.11.6 Indice generale

L'etichetta *TOC* consente di accedere alla definizione dell'indice generale, che in particolare prevede l'indicazione del livello di dettaglio che deve avere e consente la specificazione di intestazione e piè pagina differenti dal resto del documento.

Figura 54.201. Definizione dell'indice generale e della numerazione delle sezioni.

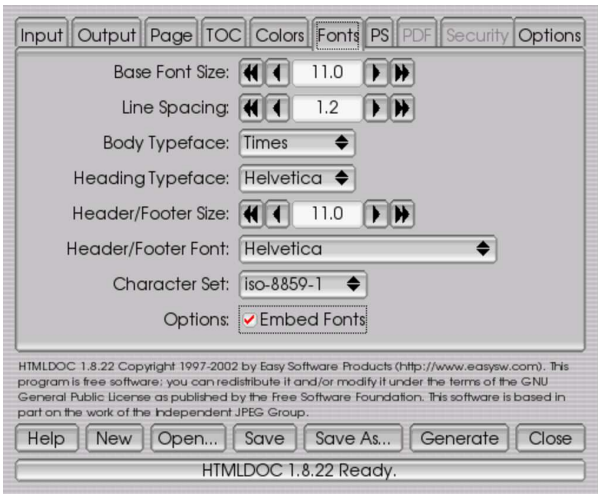


Il livello di dettaglio dell'indice si definisce anche con l'opzione '**`--toclevels` *n***', l'intestazione e il piè di pagina si possono definire con le opzioni '**`--tocheader` *xyz***' e '**`--tocfooter` *xyz***'. Infine, nella maschera si può vedere la voce NUMBERED HEADINGS, corrispondente all'opzione '**`--numbered`**', con cui si ottiene la numerazione delle sezioni (gli elementi '**`Hn`**').

54.11.7 Carattere da stampa

Sempre nel caso di composizione per la stampa, l'etichetta *Fonts* consente di definire il tipo e la dimensione dei caratteri da usare per il corpo, le sezioni, l'intestazione e il piè pagina. Per la necessità di essere compatibili al massimo, sono disponibili solo i tipi Times, Helvetica e Courier. Si osservi che l'altezza delle righe viene espressa in rapporto rispetto all'altezza dei caratteri, dove per esempio '**`1.2`**' rappresenta il 120 %.

Figura 54.202. Definizione del carattere da stampa.

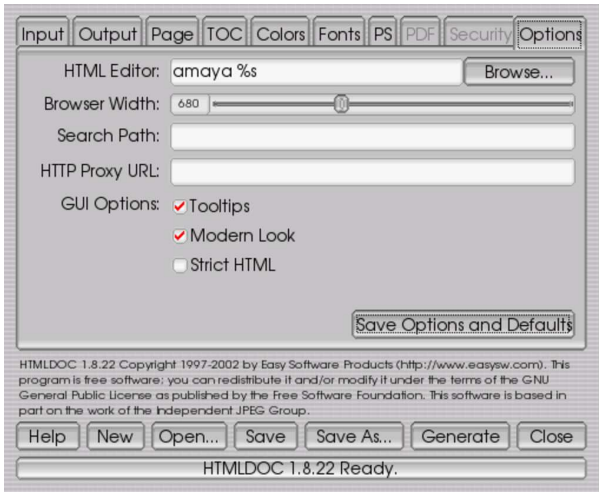


Le opzioni corrispondenti sono: '**`--bodyfont` *carattere***', '**`--fontsize` *n***' e '**`--fontspacing` *n***' per definire il carattere, la dimensione in punti e l'altezza delle righe nel testo normale; '**`--headfootfont` *carattere***' e '**`--headfootsize` *n***', per indicare il tipo di carattere e la dimensione in punti dell'intestazione e del piè di pagina; '**`--headingfont` *carattere***' per definire il tipo di carattere dei titoli delle sezioni.

54.11.8 Altre opzioni

L'ultima etichetta, *Options*, riguarda le opzioni rimanenti che non hanno trovato una collocazione migliore. In questo contesto viene definito in particolare il programma da utilizzare per la modifica dei file HTML del sorgente. Si può usare un programma per la modifica dei file di testo, oppure un applicativo più elaborato, specifico per i file HTML.

Figura 54.203. Definizione delle opzioni rimanenti.



54.11.9 Programmazione della composizione

HTMLDOC, quando funziona in modo interattivo, consente di conservare in un file l'impostazione, ovvero la configurazione di un documento, memorizzando tutte le opzioni selezionate. Quanto mostrato nelle figure di esempio, si tradurrebbe nel contenuto seguente:

```
#HTMLDOC 1.8.22
-t ps2 -f prova.ps --book --toclevels 3 --no-numbered
--toctitle "Indice generale" --title
--linkstyle underline --size A4 --left 3cm --right 3cm
--top 3cm --bottom 3cm --header .t. --footer h.1 --nup 1
--tocheader .t. --tocfooter ..i --portrait
--color --no-pscommands --no-xrxcomments --compression=1
--jpeg=0 --fontsize 11.0 --fontspacing 1.2
--headingfont Helvetica --bodyfont Times
--headfootsize 11.0
--headfootfont Helvetica --charset iso-8859-1
--browserwidth 680
uno.html due.html tre.html
```

Da *htmldoc(1)* si intuisce che l'autore suggerisca di usare l'estensione '`.book`' per questi file; tuttavia HTMLDOC non propone alcunché.

L'osservazione di questi file consente anche di capire meglio l'uso delle opzioni della riga di comando. In questo caso, volendo usare l'eseguibile '**`htmldoc`**' in modo non interattivo per fare la stessa cosa, il comando avrebbe potuto essere quello seguente:

```
$ htmldoc -t ps2 -f prova.ps --book --toclevels 3
        --no-numbered --toctitle "Indice generale"
        --title --linkstyle underline --size A4
        --left 3cm --right 3cm --top 3cm
        --bottom 3cm --header .t. --footer h.1
        --nup 1 --tocheader .t. --tocfooter ..i
        --portrait --color --no-pscommands
        --no-xrxcomments --compression=1 --jpeg=0
        --fontsize 11.0 --fontspacing 1.2
        --headingfont Helvetica --bodyfont Times
        --headfootsize 11.0 --headfootfont Helvetica
        --charset iso-8859-1 --browserwidth 680
        uno.html due.html tre.html[Invio]
```

Eccezionalmente, l'eseguibile '**`htmldoc`**' può ricevere come argomento il nome di uno di questi file '`.book`', ottenendo un funzionamento interattivo, partendo dalla configurazione contenuta nel file stesso. Supponendo di avere salvato quella configurazione nel file '`prova.book`', basterebbe riavviare nel modo seguente:

```
$ htmldoc prova.book [Invio]
```

Tabella 54.205. Riepilogo delle opzioni più comuni.

Opzione	Descrizione
`--webpage`	Il sorgente non è realizzato secondo la struttura di un libro.
`--format {ps|ps1|ps2|ps3|pdf|pdf11|pdf12|pdf13|pdf14|html}` `-t {ps|ps1|ps2|ps3|pdf|pdf11|pdf12|pdf13|pdf14|html}`	Formato finale della composizione.
`--outfile` ***file*** `-f` ***file***	File da generare dalla composizione.
`--outdir` ***directory*** `-d` ***directory***	Directory in cui inserire i file della composizione.
`--top` ***n*** `[pt|in|cm|mm]`	Margine superiore secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--bottom` ***n*** `[pt|in|cm|mm]`	Margine inferiore secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--left` ***n*** `[pt|in|cm|mm]`	Margine sinistro secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--right` ***n*** `[pt|in|cm|mm]`	Margine destro secondo l'unità di misura specificata. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--size {letter|legal|a4|universal}`	Formato della carta in base a un nome.
`--size` ***larghezza*`x`*altezza* `[pt|in|cm|mm]`	Formato della carta espresso esattamente. Se non si specifica l'unità di misura, si fa riferimento a punti tipografici ('**pt**').
`--duplex` `--no-duplex`	Richiede che il formato generato sia o non sia predisposto per la stampa su entrambi i lati del foglio.
`--header` ***xyz*** `--footer` ***xyz***	Intestazione e piè di pagina.
`--tocheader` ***xyz*** `--tocfooter` ***xyz***	Intestazione e piè di pagina dell'indice generale.
`--headfootfont {courier|times|helvetica|symbol}`	Carattere da usare nelle intestazioni e nei piè di pagina.
`--headfootsize` ***n***	Dimensione del carattere da usare nelle intestazioni e nei piè di pagina.
`--headingfont {courier|times|helvetica|symbol}`	Carattere da usare nei titoli delle sezioni.

Opzione	Descrizione
`--bodyfont {courier|times|helvetica|symbol}` `--textfont {courier|times|helvetica|symbol}`	Carattere da usare nel corpo.
`--fontsize` ***n***	Dimensione del carattere normale in punti.
`--fontspacing` ***n***	Altezza della riga rispetto alla dimensione del carattere.
`--logoimage` ***file***	Definisce il file da usare come logo.
`--bodyimage` ***file***	Definisce il file da usare come sfondo di tutte le pagine.
`--titlefile` ***file*** `--titleimage` ***file***	File da usare per la copertina (può essere anche un'immagine).
`--book`	Specifica che il sorgente HTML è strutturato in modo appropriato alla composizione di un libro.
`--charset iso-8859-`***n***	Specifica la codifica da usare, secondo la codifica ISO 8859-***n***. Sono comunque disponibili anche altre codifiche espresse secondo parole chiave differenti.
`--no-toc`	Non genera l'indice generale.
`--toclevels` ***n***	Definisce il numero di livelli dell'indice generale.
`--toctitle "`***titolo***`"`	Definisce il titolo dell'indice generale.
`--title` `--no-title`	Genera o non genera la copertina.
`--embedfonts` `--no-embedfonts`	Richiede o meno l'inclusione di tutte le informazioni sui caratteri tipografici usati per la composizione PostScript o PDF.
`--links` `--no-links`	Genera o non genera i riferimenti ipertestuali nella composizione in formato PDF.
`--linkstyle {plain|underline}`	Dichiara lo stile con cui evidenziare i riferimenti ipertestuali.
`--numbered` `--no-numbered`	Numera o non numera le sezioni del documento.
`--nup {1|2|4|6|9|16}`	Mette assieme più pagine virtuali per ogni pagina fisica.

La tabella 54.205 riassume le funzionalità delle opzioni principali di HTMLDOC quando viene usato in modo non interattivo. È importante tenere in considerazione alcune cose, che vengono descritte brevemente nel seguito.

È possibile gestire solo immagini in formato GIF, JPG e PNG; in particolare, quando si ottiene una conversione in HTML, si usano le stesse immagini di partenza.

La dimensione della carta può essere indicata per nome, oppure direttamente, fornendo larghezza e altezza, nella forma ***m***`x`***n***, dove ***m*** rappresenta la larghezza e ***n*** l'altezza. Si osservi che questi due numeri vanno completati con l'indicazione finale dell'unità di misura (come si vede nella tabella), una volta sola per entrambi i valori. Il formato speciale denominato '**`universal`**' si riferisce a un compro-

messo tra il formato A4 e il formato lettera (8,5 in × 11 in), in modo che possa essere stampato con entrambi i tipi di carta; in pratica si utilizza la larghezza del formato A4 e l'altezza del formato lettera.

Le opzioni che definiscono il contenuto delle intestazioni e dei piè di pagina, utilizzano una simbologia speciale, in cui tre lettere indicano rispettivamente la parte sinistra, quella centrale e quella destra della riga. La tabella 54.206 riepiloga questi simboli.

Tabella 54.206. Simboli usati negli argomenti delle opzioni che definiscono il contenuto delle intestazioni e dei piè di pagina.

Simbolo	Descrizione
`.`	Vuoto.
`:`	Numero della pagina del capitolo in relazione alla quantità totale di pagine del capitolo, secondo la forma: '***n_pagina*/*n_complessivo***'.
`/`	Numero della pagina del documento in relazione alla quantità totale di pagine del documento, secondo la forma: '***n_pagina*/*n_complessivo***'.
`t`	Titolo del documento.
`c`	Capitolo attuale.
`h`	Sezione attuale.
`d`	Data.
`T`	Orario.
`D`	Data e orario.
`l`	Logo.
`1`	Pagina del documento in numero arabo normale.
`a`	Pagina del documento in lettere minuscole.
`A`	Pagina del documento in lettere maiuscole.
`i`	Pagina del documento in numero romano minuscolo.
`I`	Pagina del documento in numero romano maiuscolo.
`C`	Pagina del capitolo in numero arabo normale.

54.11.10 Informazioni particolari nel sorgente

«

All'interno del sorgente HTML, alcune informazioni vengono trattate in modo speciale da HTMLDOC. Si tratta in special modo di commenti SGML che acquistano il significato di direttive, che spesso vanno a sovrapporsi a opzioni della riga di comando di HTMLDOC. La tabella 54.207 successiva descrive alcune di queste direttive.

Tabella 54.207. Alcune direttive in forma di commenti SGML.

Direttiva	Descrizione
`<-- MEDIA SIZE "`***dimensione***`" -->`	Dimensioni della pagina. Può trattarsi delle parole chiave '**`letter`**', '**`legal`**', '**`universal`**', '**`A4`**', oppure delle dimensioni espresse secondo la forma '***m*x*n*[pt|in|cm|mm]**'.

`<-- MEDIA TOP `***n***`[pt|in|cm|mm] -->`
`<-- MEDIA BOTTOM `***n***`[pt|in|cm|mm] -->`
`<-- MEDIA LEFT `***n***`[pt|in|cm|mm] -->`
`<-- MEDIA RIGHT `***n***`[pt|in|cm|mm] -->`
	Margine: superiore, inferiore, sinistro, destro.

`<-- MEDIA LANDSCAPE YES|NO -->`	Abilita o disabilita la composizione orizzontale.
`<-- MEDIA DUPLEX YES|NO -->`	Abilita o disabilita la composizione adatta per la stampa fronte-retro.

Direttiva	Descrizione

`<-- HEADER LEFT "`***testo***`" -->`
`<-- HEADER CENTER "`***testo***`" -->`
`<-- HEADER RIGHT "`***testo***`" -->`
	Parte sinistra, centrale e destra del testo da mettere come intestazione della pagina attuale.

`<-- FOOTER LEFT "`***testo***`" -->`
`<-- FOOTER CENTER "`***testo***`" -->`
`<-- FOOTER RIGHT "`***testo***`" -->`
	Parte sinistra, centrale e destra del testo da mettere alla base della pagina attuale.

`<-- HALF PAGE -->`	Salta alla prossima mezza pagina.
`<-- NEW PAGE -->` `<-- BREAK PAGE -->`	Salta all'inizio della prossima pagina.
`<-- NEED `***n_righe***` -->` `<-- NEED `***n***`{pt|in|cm|mm} -->`	Salta all'inizio della prossima pagina se non è disponibile lo spazio verticale richiesto.
`<-- NUMBER-UP {1|2|4|6|9|16} -->`	Mette assieme più pagine virtuali per ogni pagina fisica.

Per comporre il testo delle intestazioni e dei piè di pagina, si possono usare dei «simboli» speciali, che assomigliano a variabili. Si tratta precisamente di nomi prefissati da un dollaro ('**$**'), che vengono espansi in fase di composizione. La tabella 54.208 descrive alcuni di questi simboli.

Tabella 54.208. Alcuni simboli usati nelle intestazioni e nei piè di pagina, dichiarati attraverso le direttive speciali in forma di commenti SGML.

Simbolo	Descrizione
`$$`	Rappresenta un dollaro singolo.
`$TITLE`	Inserisce il titolo del documento.
`$CHAPTER`	Inserisce il titolo della sezione corrispondente al capitolo corrente.
`$HEADING`	Inserisce il titolo della sezione corrente.
`$CHAPTERPAGE` `$CHAPTERPAGE(1|i|I|a|A)`	Inserisce il numero di pagina del capitolo, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**`1`**' numero arabo normale; '**`i`**' numero romano minuscolo; '**`I`**' numero romano maiuscolo; '**`a`**' sequenza alfabetica minuscola; '**`A`**' sequenza alfabetica maiuscola.
`$CHAPTERPAGES` `$CHAPTERPAGES(1|i|I|a|A)`	Inserisce il numero complessivo di pagine del capitolo, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**`1`**' numero arabo normale; '**`i`**' numero romano minuscolo; '**`I`**' numero romano maiuscolo; '**`a`**' sequenza alfabetica minuscola; '**`A`**' sequenza alfabetica maiuscola.
`$PAGE` `$PAGE(1|i|I|a|A)`	Inserisce il numero di pagina assoluto, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**`1`**' numero arabo normale; '**`i`**' numero romano minuscolo; '**`I`**' numero romano maiuscolo; '**`a`**' sequenza alfabetica minuscola; '**`A`**' sequenza alfabetica maiuscola.

Simbolo	Descrizione
`$PAGES`	Inserisce il numero complessivo di pagine assoluto del documento, eventualmente secondo il formato stabilito dalla lettera tra parentesi: '**1**' numero arabo normale; '**i**' numero romano minuscolo; '**I**' numero romano maiuscolo; '**a**' sequenza alfabetica minuscola; '**A**' sequenza alfabetica maiuscola.
`$DATE`	Inserisce la data.
`$TIME`	Inserisce l'ora.
`$LOGOIMAGE`	Inserisce soltanto il logo, ignorando il resto del testo.

HTMLDOC trae delle informazioni da alcuni elementi '**META**' comuni. La tabella 54.209 descrive alcuni tipi riconosciuti e il significato loro attribuito.

Tabella 54.209. Elementi '**META**' riconosciuti da HTMLDOC.

Sintassi	Descrizione
`<META NAME="AUTHOR" CONTENT="`***autore***`">`	Autore del documento.
`<META NAME="COPYRIGHT" CONTENT="`***copyright***`">`	Copyright del documento.
`<META NAME="GENERATOR" CONTENT="`***applicazione***`">`	Applicazione che ha generato il sorgente HTML.
`<META NAME="KEYWORDS" CONTENT="`***parole_chiave***`">`	Elenco delle parole chiave più importanti, che riconducono ai contenuti del documento.
`<META NAME="SUBJECT" CONTENT="`***genere***`">`	Genere a cui appartiene il documento.

54.12 Motori di ricerca e robot

Più passa il tempo e più sono i documenti che vengono pubblicati su Internet. I motori di ricerca, ovvero i servizi che gestiscono gli indici delle pubblicazioni, sono sempre più sommersi di lavoro. In questa situazione, ognuno applica una propria politica di filtro dei documenti che vengono sottoposti per l'inclusione nel loro indice. In generale, non basta realizzare un documento HTML corretto, occorre pensare anche ai motori di ricerca.

Il documento HTML, per poter essere preso in considerazione in modo corretto dai motori di ricerca, deve avere una serie di elementi '**META**' nell'intestazione, contenenti alcune informazioni salienti. Ciò permette la classificazione del documento e la creazione di indici chiari per l'utente di quel servizio. Tuttavia, il problema è che non tutti i motori di ricerca utilizzano le stesse informazioni nello stesso modo; così, ci si affida generalmente all'esperienza degli altri per la compilazione di tali elementi. Qui si raccolgono solo alcune indicazioni che, però, potrebbero essere superate facilmente con il passare del tempo.

54.12.1 Elementi META

Gli elementi '**META**' sono vuoti, nel senso che non delimitano alcun testo, e si collocano nell'intestazione del file HTML, ovvero nell'elemento '**HEAD**'. Nella maggior parte dei casi, l'elemento '**META**' si utilizza con l'attributo '**name**' e l'attributo '**content**', attraverso i quali si stabilisce un nome a cui viene assegnato un contenuto.

Il DTD dell'HTML non stabilisce quali siano i nomi che si possono usare per l'attributo '**name**' e da questo nascono tutti i problemi. In particolare, c'è da considerare che alle volte i nomi e i valori abbinati non fanno differenza tra maiuscole e minuscole, altre volte pare che la facciano.

L'esempio seguente mostra un caso tipico di utilizzo per un documento realizzato in italiano:


```
<!DOCTYPE HTML PUBLIC "ISO/IEC 15445:2000//DTD HTML//EN">
<HTML LANG="it">
<HEAD>
    <META HTTP-EQUIV="Content-Type"
          CONTENT="text/html; charset=UTF-8">
    <META NAME="generator" CONTENT="ALtools">
    <META NAME="description"
          CONTENT="GNU/Linux e il software libero"
          LANG="it">
    <META NAME="description"
          CONTENT="GNU/Linux and free software" LANG="en">
    <META NAME="keywords"
          CONTENT="GNU/Linux, Unix, software, software
                   libero, free software">
    <META NAME="distribution" CONTENT="Global" LANG="en">
    <META NAME="rating" CONTENT="General" LANG="en">
    <META NAME="resource-type" CONTENT="document" LANG="en">
    <META NAME="classification" CONTENT="computers"
          LANG="en">
    <META NAME="revisit-after" CONTENT="7 days"
          LANG="en">
    <META NAME="ROBOTS" CONTENT="ALL">
    <META NAME="SPIDERS" CONTENT="ALL">
    <META NAME="author" CONTENT="Daniele Giacomini">
    <META NAME="copyright"
          CONTENT="© 1997-2001 Daniele Giacomini">
    <TITLE>Appunti Linux</TITLE>
    ...
</HEAD>
...
```

Il significato di queste informazioni dovrebbe essere intuitivo, salvo qualche caso, ma in particolare è necessario osservare un problema: alcune cose sono espresse attraverso sigle o parole chiave che hanno significato per la lingua inglese, mentre potrebbero essere attese parole o definizioni diverse nel caso di un documento in italiano. Nell'esempio si può osservare che l'elemento '**HTML**' possiede l'attributo '**lang**' a cui è assegnato il valore '**it**', allo scopo di indicare che tutto il documento è scritto in lingua italiana. Pertanto, per modificare questo assunto negli elementi '**META**' in cui il linguaggio può avere importanza, è stato aggiunto nuovamente l'attributo '**lang**' con il valore '**en**'. Può darsi che questa precauzione non serva a nulla, ma potrebbe essere importante in futuro.

Eventualmente, si potrebbe anche arrivare a duplicare alcune informazioni per diversi linguaggi. Per esempio, l'informazione denominata '**description**' viene fornita due volte: prima in italiano e poi in inglese.

L'elenco seguente descrive brevemente le informazioni più importanti che si possono dare in questo modo.

Valore assegnato	Descrizione
`description` `Description`	Si tratta di una descrizione breve del contenuto che potrebbe essere mostrato negli indici. A titolo indicativo, non dovrebbe superare le 25 parole, per essere certi che sia presa in considerazione integralmente.
`keywords` `Keywords`	Si tratta di un elenco di parole, o frasi brevi, separate da una virgola. Queste parole rappresentano gli argomenti principali del documento. Indicandole in questo modo, si cerca di farle risaltare (anche se nel documento vengono usate poco o non vengono usate affatto), per far sì che vengano prese in considerazione in modo particolare. A titolo indicativo, l'elenco non dovrebbe superare le 25 parole, per essere certi che questo venga preso in considerazione. Si intuisce che le prime parole di questo elenco devono essere le più importanti.
`distribution` `Distribution`	Probabilmente si riferisce all'estensione che ha o può avere la diffusione del documento. Le parole che possono essere assegnate sono '**Global**' e '**Local**', con i significati che si possono intuire.

Valore assegnato	Descrizione
`rating` `Rating`	Probabilmente si riferisce al tipo di pubblico a cui si rivolge il documento. In generale viene assegnata solo la parola chiave '**`General`**'; qualcuno suggerisce anche l'uso di, '**`Mature`**' e '**`Restricted`**', ma il significato in pratica non è chiaro.
`classification` `Classification`	Si tratta della classificazione del contenuto del documento. È difficile fare un elenco dei termini che si possono usare, perché dipendono dal motore di ricerca. Probabilmente si può trattare di: '**`business`**', '**`computers`**', '**`entertainment`**', '**`internet`**', '**`miscellaneous`**', '**`personal`**'.
`resource-type`	Si tratta della definizione che si dà al documento HTML. Da quanto si vede, si usa sempre solo la parola chiave '**`document`**' (solo in minuscolo).
`revisit-after`	Apparentemente, questa indicazione serve a richiedere al motore di ricerca di ripassare dopo un certo numero di giorni. Non è garantito il successo di questa richiesta, ma nulla vieta di provarci.
`ROBOTS`	Questa informazione serve a chiedere esplicitamente o a vietare la scansione e l'indicizzazione. In generale si assegna la parola chiave '**`ALL`**' perché venga preso in considerazione il documento a tutti gli effetti, assieme ai riferimenti a cui punta, mentre si usa la parola chiave '**`INDEX`**' per richiedere la sola indicizzazione e '**`FOLLOW`**' per seguire i riferimenti. Per evitare l'indicizzazione si usa '**`NOINDEX`**', mentre per evitare di seguire i riferimenti si usa '**`NOFOLLOW`**'. Qualcuno suggerisce di utilizzare la stringa '**`ALL,INDEX,FOLLOW`**' per ottenere il risultato migliore.
`SPIDERS`	Apparentemente funziona nello stesso modo di '**`ROBOTS`**' e probabilmente accetta gli stessi valori.

54.12.2 Filtro iniziale alla scansione dei robot

Nel momento in cui si è posto il problema dell'esistenza di tutta una serie di servizi di scansione della documentazione su Internet, si è pensato all'opportunità di bloccare, in certe circostanze, il lavoro di questi «robot». Gli amministratori dei servizi HTTP hanno la possibilità di realizzare il file '`/robots.txt`', contenente l'indicazione dei percorsi che non devono essere scanditi.

Anche se si tratta di un compito che riguarda gli amministratori, è opportuno sapere leggere le istruzioni di questo file, nel caso esista, per sapere se il proprio documento può essere raggiunto o meno dai motori di ricerca e da altri servizi simili.

Il file in questione, collocato all'inizio della gerarchia del servizio HTTP a cui si riferisce, è un file di testo normale, in cui si indicano dei commenti, preceduti dal simbolo '`#`', e una serie di campi nella forma:

```
campo: valore
```

Le informazioni di questo file sono suddivise in base al nome del programma robot che si vuole filtrare:

```
User-agent: nome
```

Uno o più campi del genere, posti di seguito, iniziano la definizione del filtro riferito ai programmi rispettivi. Se al posto del nome si indica un asterisco, si intendono simultaneamente tutti i programmi che non siano stati presi in considerazione diversamente.


```
Disallow: [percorso]
```

Il campo '**`Disallow`**' serve a specificare un percorso da escludere dalla scansione dei robot presi in considerazione.[8]

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow /tmp/
Disallow /cgi-bin/
Disallow /prova.html
```

Supponendo che l'esempio si riferisca al file '`http://www.brot.dg/robots.txt`', si mostra il caso in cui si vogliono escludere tutti i robot dal contenuto di '`http://www.brot.dg/tmp/`', '`http://www.brot.dg/cgi-bin/`' e dal file '`http://www.brot.dg/prova.html`'.

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow
```

In questo caso non si esclude alcunché.

```
# http://www.brot.dg/robots.txt
User-agent: *
Disallow /
```

Questo nuovo esempio esclude l'accesso a tutto il servizio.

54.13 Riferimenti

- T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, *RFC 2396: Uniform Resource Identifiers (URI): General Syntax*, 1998, http://www.ietf.org/rfc/rfc2396.txt
- Wikipedia, *Uniform resource name*, http://en.wikipedia.org/wiki/Uniform_resource_name
- *Using DNS as a URN resolver*, http://stackoverflow.com/questions/5476121/using-dns-as-a-urn-resolver
- International ISBN agency, *The ISBN Users' Manual*, http://www.isbn.org/standards/home/isbn/International/ISBNmanual.asp
- W3C, *MarkUp Validation Service*, http://validator.w3.org/
- W3C, *Link Checker*, http://validator.w3.org/checklink
- *ISO/IEC 15445:2000*, http://www.scss.tcd.ie/misc/15445/15445.html , https://www.scss.tcd.ie/misc/15445/15445.HTML
- Wikipedia, *HTML5*, http://it.wikipedia.org/wiki/HTML5
- *Character sets*, http://www.iana.org/assignments/character-sets
- W3C, *Technical Reports and Publications*, http://www.w3.org/TR/
  - *HTML 4.01 Specification*, http://www.w3.org/TR/html401/
  - *XHTML 1.0: The Extensible HyperText Markup Language*, http://www.w3.org/TR/xhtml1/
  - *Cascading Style Sheets*, http://www.w3.org/TR/REC-CSS1 , http://www.w3.org/TR/CSS2/
- FunctionX, *JavaScript tutorial*, http://www.functionx.com/javascript/
- Netscape, *Client-Side JavaScript Guide*, http://docs.oracle.com/cd/E19957-01/816-6408-10/contents.htm
- Michele Sciabarrà, *Linux e programmazione web*, McGraw-Hill, 2001, ISBN 88-386-4177-3 (in particolare il capitolo *JavaScript*)
- Jan Kärrman, *Using html2ps*
- Irène Vatton, *Amaya documentation*, http://www.w3.org/Amaya/User/
- *Amaya user manual*, http://www.w3.org/Amaya/User/doc/Manual.html
- *Amaya Home Page*, http://www.w3.org/Amaya/ , http://www.w3.org/Amaya/User/BinDist.html

• *TheFreeSite.com*, *http://www.thefreesite.com*

[1] Qui viene usato il termine «directory», ma in pratica potrebbe anche non essere esattamente una directory vera e propria.

[2] Generalmente si preferisce il formato JPG, perché originariamente il formato PNG era accettato solo da alcuni programmi di navigazione, mentre il formato GIF era brevettato.

[3] Per motivi di compatibilità con i vecchi navigatori, i marcatori di questo genere vanno indicati avendo l'accortezza di lasciare uno spazio prima della barra finale; per esempio: '`<br />`'.

[4] In generale, il buon senso dovrebbe essere sufficiente per intendere quando una caratteristica viene ereditata e quando questo non può succedere.

[5] **HTML2ps** GNU GPL

[6] **Amaya** software libero con licenza speciale

[7] **HTMLDOC** GNU GPL

[8] Non è possibile indicare dei metacaratteri (caratteri jolly) nel percorso; d'altra parte ciò non avrebbe significato, dal momento che si intendono tutti i percorsi che iniziano come indicato e proseguono poi in qualunque modo.