UNIVERSITA' DEGLI STUDI DI PADOVA
FACOLTA' DI INGEGNERIA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

HTML 5 e CSS 3: UN CONNUBIO VINCENTE PER LA PROGRAMMAZIONE WEB DI DOMANI

RELATORE: CH.MO PROF. MICHELE MORO

LAUREANDO: FEDERICO TOFFANO

ANNO ACCADEMICO: 2011-2012

INDICE

Sommario

Il principale obiettivo di questa tesi è di documentare le maggiori novità introdotte dai linguaggi HTML 5 e CSS 3. Per una giusta comprensione è necessaria una discreta conoscenza di HTML 4 e CSS 2 ed inoltre essere al corrente delle basilari potenzialità di javaScript: nel prosieguo queste conoscenze si danno per assunte.

Saranno analizzati nel dettaglio i nuovi elementi di HTML 5, le proprietà di CSS 3, saranno descritte alcune funzioni JavaScript di supporto a queste due nuove tecnologie, ci saranno alcuni esempi di codice con l'intenzione di valutare alcune delle nuove funzionalità, e infine le tabelle di compatibilità dei vari elementi relative ai browser più popolari.

Per verificare se una data proprietà è supportata o no dai vari browser, si può utilizzare un servizio come *When can I use* fruibile dal sito http://caniuse.com/. Alternativamente le varie funzionalità sono consultabili in modo rapido e intuitivo dal sito web http://html5readiness.com/ tramite un'infografica dinamica.

Per vedere l'effetto degli esempi riportati si può utilizzare un editor HTML online come http://rendera.heroku.com/.

1 INTRODUZIONE

Con l'evoluzione di HTML, acronimo di Hyper Text Markup Language (linguaggio di marcatura per ipertesti) e di CSS (fogli di stile o Cascading Style Sheets), il World Wide Web si sta trasformando in un ambiente dove i documenti caricati sono associati ad informazioni e dati facilitando così l'interrogazione e l'interpretazione dei contenuti. Questa evoluzione s'indica con il termine *web semantico* coniato da Tim Berners-Lee fondatore del www assieme a Robert Cailliau.

A causa delle divergenze ideologiche tra il W3C e il WAHTWG (i due enti coinvolti nella definizione della specifica) e i diversi interessi delle aziende coinvolte nello sviluppo dei principali browser, HTML 5 adotta alcune scelte che possono sembrare discutibili ma sono frutto di sano pragmatismo.

Perché HTML 5 e CSS 3?

Quando sono state definite le precedenti versioni, il web era strettamente collegato al concetto di ipertesto: gli utenti lo utilizzavano con il principale obbiettivo di ottenere informazioni tipicamente in forma testuale. Inoltre la velocità di connessione era mediamente bassa e di conseguenza applicazioni web quali quelle odierne erano impensabili. L'obbiettivo principale era quello di pubblicare semplici documenti testuali collegati fra loro.

Negli anni successivi il numero di utenti è cresciuto molto rapidamente e questo ha permesso investimenti per aumentare le velocità di connessione e quindi lo sviluppo di applicazioni web più complesse. Le specifiche però non erano adatte a questo nuovo utilizzo del web, quindi molti sviluppatori si sono visti costretti ad aggirare le limitazioni imposte con metodologie spesso complesse e poco eleganti.

Queste due nuove tecnologie nascono per offrire un supporto a questo nuovo mondo del web sia per quanto riguarda la strutturazione del contenuto, sia per lo sviluppo di vere e proprie applicazioni web. Il World Wide Web Consortium ha annunciato che la prima versione dello standard HTML 5 è pronta per il luglio 2014. CSS 3, rispetto alle versioni precedenti, adotta un approccio modulare. Ciascun modulo quindi copre determinate specifiche, di conseguenza il ciclo di vita e l'avanzamento è diverso per ognuno di essi.

Privacy

Con l'avvenire di HTML 5 e delle novità ad esso collegate, come il Web Storage e la geolocalizzazione, il tema della privacy è sempre più sensibile. Gli utenti che utilizzeranno questi strumenti condivideranno informazioni riguardanti i loro interessi e le loro attitudini in modo nettamente superiore rispetto alle informazioni che lasciamo in rete oggi.

Flash vs HTML 5

Dopo numerose battaglie tra le due tecnologie, Adobe ha annunciato ufficialmente che non svilupperà più Flash per dispositivi mobili, Flash Player 11 sarà l'ultima versione ufficiale rilasciata per dispositivi mobili e il futuro sarà HTML 5.

Steve Jobs, in una lettera aperta intitolata “Thoughts on Flash” del 2010, spiega quali sono i motivi che hanno portato Apple ad optare per HTML 5 preannunciando un futuro senza il software di Adobe per quanto riguarda le animazioni nel campo mobile. Forse il contributo al decadimento di questa tecnologia, per quanto riguarda i dispositivi mobili, in parte è da attribuire proprio alla campagna di pubblicità negativa dettata da Apple. Basti pensare che IPhone e Ipad non supportano tale tecnologia. Inoltre anche Microsoft, YouTube, Amazon e molte altre aziende stanno migrando verso la nuova tecnologia HTML.

Oltre alla pubblicità negativa, HTML 5, rispetto a Flash, offre un linguaggio per le animazioni molto più leggero ed è supportato da tutti i browser moderni. Attualmente HTML 5 non è ancora in grado di rimpiazzare completamente Flash ma ha il vantaggio rendere la Rete libera e non dipendente da tecnologie proprietarie.

Probabilmente in futuro, anche tramite l’utilizzo di librerie JavaScript, HTML 5 sarà in grado di rimpiazzare completamente Flash.

JavaScript

Durante questa fase di transazione rivolta verso le nuove specifiche, si sono diffuse alcune librerie JavaScript per dare un supporto alle nuove tecnologie:

- Testare la capacità o meno dei vari browser di elaborare correttamente le nuove funzionalità: *Modernizr* è una di queste, lo scopo è verificare il funzionamento di determinate funzioni indipendentemente dal browser utilizzato.
- Sostituire le funzionalità del browser mancanti con altre funzioni: la libreria *excanvas* creata da Google, ad esempio, permette l’utilizzo dell’elemento `<canvas>` sulle versioni di Internet Explorer precedenti alla 9.
- Effetti grafici: queste librerie evidenziano le potenzialità di questa innovazione, un esempio potrebbe essere la libreria *close-pixelate* di David De Sandro.

Retrocompatibilità

Innanzitutto per utilizzare le nuove specifiche bisogna adottare una strategia di retrocompatibilità nel caso in cui queste non siano supportate.

Una strategia potrebbe essere quella di identificare il browser utilizzato e la sua versione nel momento in cui si visualizza una pagina web e prendere decisioni in funzione del risultato.

Una strategia migliore prevede l’utilizzo della libreria JavaScript *Modernizr* che esegue rapidamente una serie di test per stabilire se una data funzionalità sia supportata o meno.

Test retrocompatibilià con Modernizr

```
<script>
        if(Modernizr.video()) {
                /*supportato*/
        }
        else {
                /*non supportato: strategie alternative*/
        }
</script>
```

Video è una proprietà booleana dell’oggetto Modernizr, se il suo valore è `true` allora il browser supporterà l’elemento `<video>`.

2 Semantica

Con semantica di un tag s'intende il significato che l'elemento attribuisce al suo contenuto. Con le nuove specifiche di HTML 5 gli attributi stilistici vengono gestiti solamente tramite fogli di stile in modo tale da separare il contenuto di un documento dalla sua visualizzazione.

Doctype

Il *doctype* è il primo elemento che dovrebbe comparire in un documento HTML. La sua funzione è di validare il documento e, per alcuni browser, serve ad abilitare la modalità di interpretazione standard.

Doctype HTML 5

```
<!DOCTYPE html>
```

Doctype HTML 4.01 strict

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

Inserire il *doctype* di HTML 4 come si può ben vedere non era per niente semplice. Fortunatamente con la nuova specifica la sintassi di questo elemento è molto più snella.

Codifica dei caratteri

Nell'intestazione di un documento HTML (all'interno dei tag `<head></head>`) sono definiti i tag `<meta>` tra i quali uno è utilizzato per definire la codifica dei caratteri.

Codifica caratteri HTML 5

```
<META charset=UTF-8">
```

Codifica caratteri HTML 4.01

```
<META http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Anche in questo caso si può rappresentare lo stesso attributo con una sintassi più agile.

Elementi di struttura

Gli *elementi di struttura* sono i tag che specificano la suddivisione in aree logiche di una pagina web.

Inizialmente la struttura di una pagina web era composta di tabelle suddivise in righe e colonne nelle quali erano presenti le varie sezioni del documento HTML. I principali svantaggi di questa metodologia riguardavano: i *motori di rendering,* poiché la presenza di più tabelle annidiate tra loro rallentava lo scaricamento della pagina web, e il problema della mescolanza tra elementi usati per marcare il contenuto e quelli usati per la visualizzazione. Inoltre per eseguire una sostanziale modifica sulla visualizzazione della pagina i tempi non erano di certo brevi, data la complessità della struttura che spesso veniva a crearsi.

In seguito le tabelle vennero sostituite dai `<div>`, contenitori generici, che consentono la creazione di pagine web meno complesse e di facile manutenibilità. Anche in questo caso però lo stesso elemento era finalizzato a descrivere contenuto e visualizzazione. Per marcare

il contenuto spesso si utilizzavano le classi di stile (un classico esempio è `<div class="header">`).

Sulla base di queste esperienze HTML 5 introduce nuovi elementi per poter migliorare la semantica dei documenti.

Elemento <header>

La specifica impone che questo elemento debba delimitare l'intestazione di una sezione del documento. Le intestazioni possono essere molteplici all'interno di un documento. Si pensi ad esempio un sito web di un laboratorio universitario: nell'home page ci sarà un'intestazione contenente il nome del laboratorio, un menù di navigazione e qualche altra informazione. Sotto l'intestazione potrebbe esserci una sezione contenente le recenti pubblicazioni delle ricerche effettuate dal laboratorio, ognuna di queste dovrà avere anch'essa un'intestazione che potrebbe contenere il titolo della ricerca, l'autore e la data di pubblicazione.

Esempio <header>

```
<header>
        <h1>Algoritmo Pippo</h1>
        <p>Scritto da Marco Rossi</p>
        <p>Pubblicato il 29-05-2012</p>
</header>
```

Elemento <nav>

Come già detto, nell'intestazione di una sezione possono essere presenti degli elementi con il fine di abilitare la navigazione del sito web, spesso sono inseriti anche a piè di pagina. Questi elementi dovrebbero essere racchiusi tra i tag `<nav>` ma non necessariamente qualsiasi gruppo di elementi di navigazione dev'essere delimitato da questo nuovo tag.

Esempio <nav>

```
<nav>
        <ul>
                <li><a href="ricerche.html">Ricerche</a></li>
                <li><a href="chisiamo.html">Chi siamo</a></li>
                <li><a href="contatti.html">Contatti</a></li>
        </ul>
</nav>
```

Elemento <section>

Questo elemento, secondo la definizione presente nella specifica HTML 5, rappresenta una sezione generica di un documento o applicazione. In questo contesto, individua un raggruppamento tematico di contenuti, ed in genere contiene un titolo introduttivo. Esempi d'uso potrebbero essere le sezioni di una homepage o i capitoli di un libro. Esso, comunque, non è da intendersi come un elemento la cui finalità è l'attribuzione di un determinato stile per gli elementi che racchiude, per questo scopo è sempre valido il classico `<div>`. In oltre se l'area da racchiudere è destinata a un utilizzo via RSS questo non è l'elemento appropriato per tale impiego, il tag `<article>` è più opportuno.

Elemento <article>

L'elemento <article> dev'essere utilizzato quando s'inserisce informazioni indipendenti dal resto del documento o della sezione in cui è racchiuso. Può trattarsi ad esempio di un articolo di giornale o una pubblicazione in un blog o un qualsiasi altro elemento indipendente dal contenuto.

Elemento <aside>

Se si vogliono aggiungere informazioni aggiuntive a un elemento, questo è il tag appropriato. Le informazioni che racchiude, se rimosse, non devono incidere negativamente nella

completezza dell'informazione contenuta nell'elemento a cui è associato. Può essere utilizzato ad esempio per racchiudere elementi `<nav>` aggiuntivi, barre laterali o annunci.

Elemento <footer>

Analogamente alle intestazioni anche i piè di pagina possono essere molteplici all'interno di una pagina web. Contrariamente a quanto si potrebbe pensare non è detto che questo tag debba essere l'ultimo elemento di una determinata sezione, anche se questo è quanto accade nella maggior parte dei casi.

Esempio <footer>

```
<footer>
        <nav>
                <ul>
                        <li><a href="ricerche.html">Ricerche</a></li>
                        <li><a href="chisiamo.html">Chi siamo</a></li>
                        <li><a href="contatti.html">Contatti</a></li>
                </ul>
        </nav>
        <p>Università degli studi di Padova 2012</p>
</footer>
```

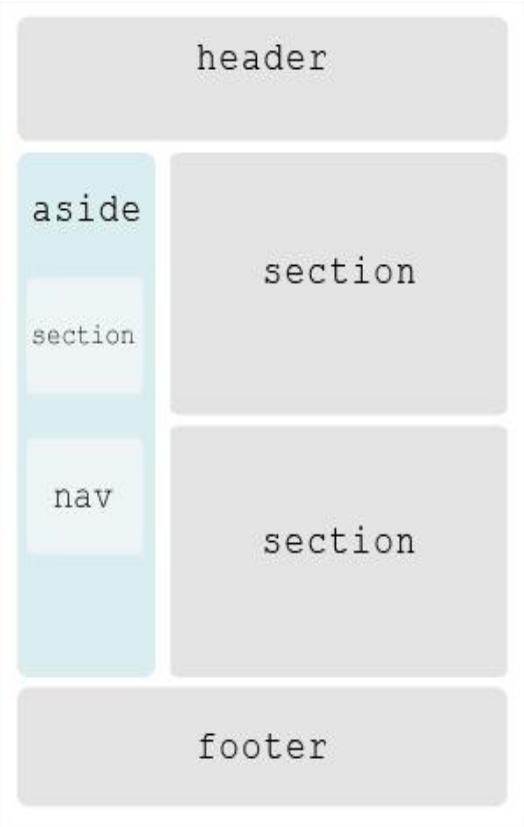

Figura 1: struttura di una pagina

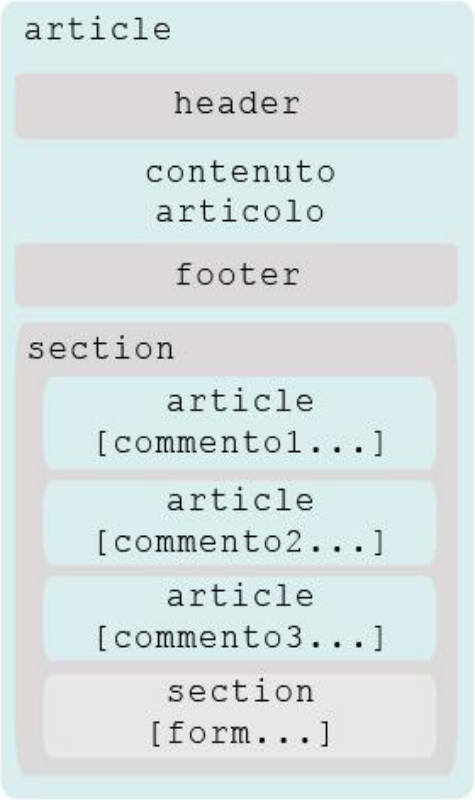

Figura 2: suddivisione semantica degli articoli

Retrocompatibilità

Le versioni di Internet Explorer precedenti alla 9 quando individuano elementi sconosciuti non applicano gli stili associati. Per aggirare questo problema si può creare in memoria questi elementi con l'utilizzo di JavaScript.

Elementi di struttura in memoria

```
<script>
        document.createElement("header");
        document.createElement("footer");
        document.createElement("nav");
        document.createElement("section");
        document.createElement("article");
        document.createElement("aside");
</script>
```

Esistono anche delle librerie JavaScript che eseguono queste operazioni come *html5shiv*. Dopo averla inclusa nella pagina web, tramite commenti condizionali, si abilita la libreria solamente per le versioni di Internet Explorer inferiori alla 9.

Includere html5shiv

```
<!--[if lt IE 9]>
        <script src="//html5shiv.googlecode.com/svn/trunk/html5.js">
        </script>
<![endif]-->
```

Nuovi elementi

I tag presentati finora concernevano le novità riguardanti la nuova struttura delle pagine web. Oltre a questi sono stati introdotti nuovi elementi con il fine di fornire un maggior dettaglio dei contenuti. Alcuni di questi tag oggi non sono ancora supportati dalle più recenti versioni dei browser più popolari.

Elementi <figure> e <figcaption>

Secondo la specifica l'elemento `<figure>` dovrebbe essere utilizzato per contenere elenchi di codice, immagini, diagrammi, ecc... Con la condizione che non siano parte integrante del contesto in cui sono inseriti. L'elemento `<figcaption>` ha il compito di arricchire la descrizione dell'elemento posto all'interno del tag `<figure>`.

Esempio <figure> e <figcaption>

```
<figure>
        <img src="dei.jpg" alt="Ingegneria informatica Padova">
        <figcaption>La sede di ingegneria informatica di Padova vista
        dall'esterno.</figcaption>
</figure>
```

Elemento <hgroup>

Questo elemento fa parte dell'intestazione di una sezione di una pagina, ma a differenza del tag `<header>`, raggruppa solamente elementi d'intestazione da `<h1>` a `<h6>`. La vera importanza del tag `<hgroup>` è che maschera l'outline dell'elemento padre che lo contiene; infatti, l'algoritmo dell'*outliner* riconosce come un titolo solamente l'heading con il valore più alto e considera tutti gli altri elementi sottotitoli. Questo consente di utilizzare ad esempio un `<h2>` come sottotitolo ma di escluderlo da un insieme di titoli `<h2>` associati ad una determinata sequenza logica.

Esempio <hgroup>

```
<hgroup>
        <h1>Unipd Lab</h1>
        <h2>Il laboratorio dell'università di Padova</h2>
</hgroup>
```

Elemento <time>

Questo elemento permette di rappresentare una data in un formato leggibile sia dagli utenti sia dalle macchine.

Esempio <time>

```
<p>Arrivò a Padova alle
<time datetiame=”2012-01-21T09:00+1:00”>
9 del mattino del 21 Gennaio 2012</time>
a causa di un ritardo del treno.</p>
```

L’attributo `datetime` indica data e ora nel formato ISO. Il fuso orario può anche essere omesso, in tal caso la data sarà stabilita dal software utilizzato per la visualizzazione. Si potrebbe anche omettere l’intero attributo `datetime` con la condizione di inserire all’interno dell’elemento `<time>` una data conforme al formato ISO.

Un altro attributo che si potrebbe aggiungere al tag è `pubdate`: un booleano che serve a indicare se la data riportata fa riferimento alla pubblicazione dell’articolo in cui è contenuto il tag `<time>`. Se non è presente nessun elemento `<article>` la data fa riferimento all’intero documento.

Elemento <progress>

Indica lo stato di completamento di un’attività. Un esempio d’uso potrebbe essere il caricamento di un file. Se l’intervallo di completamento è noto, allora il browser dovrebbe visualizzare una barra di completamento che indichi lo stato dell’esecuzione di una determinata attività. Nel caso in cui questo intervallo non sia noto, il browser dovrebbe visualizzare un oggetto che indichi che l’esecuzione dell’attività richiesta è in corso.

Esempio <progress>

```
<section>
        <p>Caricamento: <progress id="mioLoader" max="100"
        value="10"><span>0</span>%</progress></p>
</section>

<input id=”start” type="button" value="Start"
onclick="updateProgress()">

<script type="text/javascript">
        var progressBar = document.getElementById('mioLoader');
        var Value = 0;
        function updateProgress(newValue) {
                setTimeout('getValue()',10);
                progressBar.value = newValue;
                progressBar.getElementsByTagName('span')[0].textContent =
                newValue;
         }
        function getValue(){
                if (Value < 100){
                        Value ++;
                        updateProgress(Value)
                }
                else{
                        Value = 0;
                }
        }
</script>
```

Questo esempio comprende uno script che interagisce con l’elemento `<progress>` aggiornandone lo stato di completamento.

Elemento <meter>

Rappresenta una grandezza scalare all'interno di un intervallo noto, o un valore frazionario. Esempi d'uso potrebbero essere il Google rank o lo spazio su disco rimanente.

Esempio <meter>

```
<meter value="5" min=”0” max="8">5 GB utilizzati su 8 GB</meter>
```

In caso di mancato supporto, è opportuno inserire all'interno del tag del testo in modo tale da poter indicare comunque l'informazione.

Elemento <mark>

Rende evidente una porzione di testo all'interno di un documento. Un possibile utilizzo potrebbe essere quello di evidenziare le parole cercate da un utente in un testo.

Esempio <mark>

```
<p>Senza <mark>plug in</mark> di terze parti il web potrebbe diventare
per noi sviluppatori più democratico;
con le nuove API HTML5 non abbiamo più bisogno di diversi <mark>plug
in</mark> di terze parti che sino ad ora erano indispensabili per i
contenuti multimediali.</p>
```

Tabella di compatibilità

Nuovi tag semantici					
<header>	9.0+	3.0+	3.1+	4.0+	9.0+
<footer>	9.0+	3.0+	3.1+	4.0+	9.0+
<section>	9.0+	3.0+	3.1+	4.0+	9.0+
<article>	9.0+	3.0+	3.1+	4.0+	9.0+
<nav>	9.0+	3.0+	3.1+	4.0+	9.0+
<aside>	9.0+	3.0+	3.1+	4.0+	9.0+

Nuovi tag strutturali					
<figure>	9.0+	3.0+	3.1+	4.0+	9.0+
<figcaption>	9.0+	3.0+	3.1+	4.0+	9.0+
<hgroup>	9.0+	3.0+	3.1+	4.0+	9.0+
<time>	9.0+	3.0+	3.1+	4.0+	9.0+
<meter>	No	6.0+	5.2+	6.0+	11.0+
<progress>	10.0+	6.0+	5.2+	6.0+	10.6+
<mark>	9.0+	3.0+	3.1+	4.0+	9.0+

3 Web Forms 2.0

I form, prima della definizione della nuova specifica, utilizzavano una modalità di interazione efficace ma un po' troppo rudimentale. Le principali novità di HTML 5 dei web forms riguardano principalmente la semantica e l'aggiunta di nuovi attributi agli elementi con lo scopo di strutturare meglio la pagina e facilitarne l'accessibilità.

L'attributo type

Finora l'attributo `type` dell'elemento `<input>` supportava dieci valori (`text`, `button`, `radio`, ecc..), la nuova specifica ne supporta 23. Fortunatamente se il valore dell'attributo non è riconosciuto, sarà interpretato con il valore di default `type="text"`, quindi si potranno utilizzare i nuovi attributi anche se non supportati da tutti i browser.

Input type : search

La specifica, per questo *input type*, permette una diversa visualizzazione in funzione della piattaforma utilizzata, diversa dalla classica visualizzazione di un campo testo. Ad esempio, utilizzando Safari o Google Chrome, se scriviamo qualcosa nel campo, compare una piccola X sulla destra che, se cliccata, svuota il campo. Inoltre se Safari è utilizzato su Mac Os, il campo sarà visualizzato con i bordi arrotondati.

Esempio search

```
<form name="ricerca" method="post" action="/search">
        <label> Cerca:
                <input type="search" autocomplete="on"
                placeholder="HTML, CSS, ..."
                name="keyword" required maxlength="50">
        </label>
        <input type="submit" value="Ricerca">
</form>
```

Input type : email

L'indirizzo di posta elettronica è un'informazione spesso richiesta durante la compilazione di un form, questo *input type* sarà utilizzato per creare un campo in cui si dovrà inserire questa informazione. Opera e Google Chrome eseguono un controllo con il fine di valutare se il campo è un indirizzo email valido. Se l'indirizzo non è valido, sarà visualizzato un messaggio di errore. Inoltre i dispositivi mobili, nel momento in cui si abilita l'inserimento del testo nel campo, potrebbero modificare la tastiera, ad esempio visualizzando la chiocciola come accade con IPhone.

Esempio email

```
<form name="registrazione" method="post" action="/send">
        <label> Email:
                <input type="email" name="email" autocomplete="on"
                placeholder="email@domain.ext">
        </label>
        <input type="submit" value="Registra">
</form>
```

Questo controllo inoltre accetta l'attributo booleano `multiple` che se impostato a `true` permette l'inserimento di più indirizzi email separati da una virgola.

Input type: url

Un altro dato spesso richiesto nella compilazione di un form è l'indirizzo web. Come per il capo email Opera e Google Chrome controllano la sintassi del testo immesso nello spazio dedicato. Safari Mobile su IPhone, durante la compilazione del campo, visualizza una tastiera che include il punto, la *slash* e l'estensione `.com`.

Esempio url

```
<form name="registrazione" method="post" action="/send">
        <label> Indirizzo web:
                <input type="url" name="url" autocomplete="on"
                placeholder="http://mywebsite.com">
        </label>
        <input type="submit" value="Registra">
</form>
```

Come per il *type email* anche il *type url* supporta l'attributo booleano `multiple` per l'inserimento di più indirizzi web.

Input type: tel

Questo *input type* è stato aggiunto per permettere la per creazione di un campo adatto all'inserimento di numeri di telefono. A differenza dei campi email e url non è imposto un particolare formato perché a livello internazionale tra i numeri di telefono ci sono sostanziali differenze riguardanti la struttura. Ancora una volta IPhone offre una tastiera dedicata per questo elemento visualizzando un tastierino numerico.

Esempio tel

```
<form name="registrazione" method="post" action="/send">
        <label> Numero di telefono:
                <input type="tel" name="tel">
        </label>
        <input type="submit" value="Registra">
</form>
```

Input type: number

Questo *input type* è stato pensato per l'inserimento di valori numerici. La visualizzazione prevista di questo elemento, rispetto a un campo di testo, presenta due frecce sul lato sinistro con il fine di poter incrementare o diminuire il valore inserito. Sia Android sia iOS di IPhone offrono una tastiera alternativa per questo campo. Inoltre si possono impostare tre attributi all'interno di questo elemento:

- `min`: specifica il valore minimo dell'intervallo, il valore predefinito è 0
- `max`: specifica il valore massimo dell'intervallo, il valore predefinito è 100
- `step`: specifica il salto tra un valore ed il successivo nell'intervallo scelto

Dopo aver scelto il valore, è previsto un controllo con il fine di verificare l'appartenenza del numero inserito all'insieme preimpostato.

Esempio number

```
<form name="registrazione" method="post" action="/send">
        <label> Peso:
                <input type="number" name="peso" min="10" max="200"
                step="0.1" value="70.5">
        </label>
        <input type="submit" value="Registra">
</form>
```

Input type: range

Questo campo, simile al precedente, permette l'inserimento di numeri ma tramite l'utilizzo di una barra. La specifica stabilisce che quando s'intende utilizzare questo controllo non è importante conoscere con esattezza il valore scelto ma è sufficiente ottenere un valore indicativo. Anche questo campo prevede gli attributi *min max* e *step* descritti prima.

Esempio range

```
<form name="intervista" method="post" action="/send">
        <label> Come valuti questo sito?
                <input type="range" name="peso" min="0" max="10"
                step="1">
        </label>
        <input type="submit" value="Inoltra">
</form>
```

Input type per le date

Prima della definizione della nuova specifica per gestire le date era necessario ricorrere a supporti esterni, ora è molto più semplice grazie ai nuovi elementi di seguito descritti:

- `datetime`: gestisce sia la data che l'ora (con fuso orario);
- `datetime-local`: gestisce sia la data che l'ora (senza fuso orario);
- `date`: gestisce le date;
- `month`: gestisce i mesi;
- `week`: gestisce le settimane;
- `time`: gestisce l'ora.

Per tutti questi *input type* abbiamo degli attributi specifici che sono:

- `min`: rappresenta il minimo valore accettato;
- `max`: rappresenta il massimo valore accettato;
- `step`: rappresenta la granulosità dei valori accettati.

Esempi date

```
<label> Datetime
        <input type="datetime" name="mydatetime">
</label>
<label> Datetime-local
        <input type="datetime-local" name="mydatetime">
</label>
<label> Date
        <input type="date" name="mydatetime">
</label>
<label> Month
        <input type="month" name="mydatetime">
</label>
<label> Week
        <input type="week" name="mydatetime">
</label>
<label> Time
        <input type="time" name="mydatetime">
</label>
```

Input type: color

Lo scopo di questo elemento è di permettere la selezione di un colore in modo semplice, tramite l'utilizzo di una tabella, ottenendo una stringa esadecimale di sei cifre che rappresenta il colore in formato RGB.

Esempio color

```
<form name="intervista" method="post" action="/send">
        <label> di che colore è la tua macchina?
                <input type="color" name="mycar">
        </label>
        <input type="submit" value="Inoltra">
</form>
```

Input con datalist

Con le precedenti versioni di HTML per permettere la scelta di un'opzione da un elenco preimpostato veniva utilizzata una combinazione dei due tag `<select>` e `<option>`. Nel caso in cui si fosse dovuto dare la possibilità di immettere un'informazione non compresa nell'elenco, l'unica soluzione era aggiungere un ulteriore campo testo. Con HTML 5 si può fare entrambe le cose utilizzano un unico controllo `<input>` impostando l'attributo `list` di quest'ultimo con l'id della lista (rappresentata dal tag `<datalist>`) delle opzioni selezionabili (ognuna definita all'interno di un tag `<option>`).

Esempio input con datalist

```
<form name="intervista" method="post" action="/send">
        <label> Che sitema operativo preferisci?
                <input type="text" name="mood" list="OS">
                <datalist id="OS">
                        <option value="Ubuntu">
                        <option value="Debian">
                        <option value="Kubuntu">
                </datalist>
        </label>
        <input type="submit" value="Inoltra">
</form>
```

Per una corretta visualizzazione l'attributo `value` di `<option>` non deve essere vuoto ed inoltre `<option>` non deve contenere l'attributo `disabled`.

Nuovi attributi

Oltre agli attributi relativi ad alcuni degli *input type* descritti (`multiple`, `min`, `max`, `step` e `list`) ne sono stati introdotti altri, applicabili a tutti gli elementi `<input>`.

Attributo autocomplete

Di default il valore di questo attributo è impostato a `on`, ovvero i browser compilano i campi di un form in funzione di altre compilazioni già effettuate. Spesso risulta comoda questa funzionalità poiché velocizza il processo d'inserimento dei dati, se però i campi in questione sono relativi a dati sensibili (come le password) potrebbe convenire impostare il valore dell'attributo a `off` dato che i dati vengono memorizzati all'interno del browser con la conseguenza di una più facile accessibilità da parte di persone o software sgraditi.

Attributo autofocus

L'attributo booleano `autofocus` serve ad impostare il focus su un determinato elemento di un form dopo il caricamento di una pagina. Nel caso in cui questo attributo sia stato applicato

all'interno di più campi, il focus sarà assegnato all'ultimo elemento a cui è applicato. L'utilizzo di questo attributo all'interno di pagine lunghe potrebbe risultare scomodo in quanto la pressione della barra spaziatrice agirà sul controllo mentre alcuni utenti la utilizzano per scorrere velocemente la pagina.

Nel caso in cui `autofocus` non sia supportato, si potrebbe ottenere lo stesso effetto tramite un semplice script.

Autofocus sostitutivo con JavaScript

```
if(!Modernizr.input.autofocus){
        Document.getElementById("password").focus()
}
```

Attributo form

Prima della definizione della nuova specifica un elemento di un form doveva necessariamente essere racchiuso tra i due tag `<form></form>`. Ora se a un elemento esterno a un form è aggiunto questo attributo impostando il suo valore con l'id di uno o più form (separati da virgola), continuerà comunque ad esserne parte integrante.

Attributo pattern

Con questo attributo le espressioni regolari potranno essere gestite con HTML 5. Il suo valore dovrà quindi essere un'espressione regolare valida. All'interno dell'attributo `title` si potrebbe inserire una spiegazione del formato richiesto per la compilazione dell'elemento. Nel caso in cui i vincoli non siano rispettati, il browser blocca l'invio del form avvisando l'utente.

Attributo paceholder

Il valore testuale di questo attributo è visualizzato all'interno di un `input` o di una `textarea`. Il testo scompare nel momento in cui il campo assume il focus oppure quando l'utente lascia il focus dell'elemento senza averlo compilato. Può essere utilizzato per fornire un suggerimento all'utente riguardante la compilazione, oppure può essere utilizzato come semplice etichetta.

Attributo required

Per rendere obbligatoria la compilazione di un campo contenuto in un form è sufficiente aggiungere questo attributo booleano. Se l'utente tenta di inviare i dati senza aver compilato tutti i campi contenenti questo attributo il browser segnala l'errore.

Attributo novalidate

Questo attributo si applica all'elemento `<form>` e permette di saltare tutte le validazioni dei tag che da esso discendono.

Esempio di un form di registrazione

```
<form name="registrazione" method="post" action="/send">
	<fieldset>
		<legend>Registrati</legend>
		<p>
			<label for="nome">Nome*</label>
			<br>
			<input id="nome" type="text" name="nome" autofocus
			placeholder="inserisci nome e cognome" size="40"
			required>
		</p>
		<p>
			<label for="email">Email*</label>
			<br>
			<input id="email" type="email" name="email"
			placeholder="tua@mail.it" size="40" required>
		</p>
		<p>
			<label for="pass">Password*</label>
			<br>
			<input id="pass" type="password" name="pass"
			placeholder="almeno 5 caratteri" size="40"
			required pattern="(\s*(\S)\s*){5,}"
			autocomplete="off">
		</p>
		<p>
			<label for="telefono">Telefono</label>
			<br>
			<input id="telefono" type="tel" name="telefono"
			placeholder="inserisci un recapito telefonico"
			size="40">
		</p>
		<p>
			<input type="submit" name="registra"
		value="Registra">
		</p>
	</fieldset>
</form>
```

Ovviamente tutti i controlli eseguiti lato client è necessario rieseguirli lato server dato che i controlli html sono facili da eludere. La finalità dei controlli lato client è di migliorare la struttura e di facilitare l'accessibilità da parte degli utenti.

Tabella di compatibilità

Nuovi input type					
tel	10.0+	4.0+	5.0+	6.0+	10.6+
search	10.0+	4.0+	5.0+	6.0+	10.6+
url	10.0+	4.0+	5.0+	6.0+	10.6+
email	10.0+	4.0+	5.0+	6.0+	10.6+
range	10.0+	No	4.0+	6.0+	9.0+
number	No	No	4.0+	20.0+	11.0+
datetime	No	No	5.0+	20.0+	10.6+
color	No	No	No	20.0+	11.0+
datalist	10.0+	4.0+	No	20.0+	9.0+

Nuovi attributi					
`autocomplete`	No	4.0+	5.2+	17.0+	10.6+
`autofocus`	10.0+	4.0+	5.0+	6.0+	11.0+
`form`	10.0+	4.0+	5.2+	10.0+	10.6+
`pattern`	10.0+	4.0+	No	6.0+	10.6+
`placeholder`	10.0+	4.0+	4.0+	10.0+	11.10+
`required`	10.0+	6.0+	No	6.0+	10.6+
`min, max`	10.0+	No	5.0+	6.0+	10.6+
`multiple`	10.0+	3.6+	5.0+	6.0+	11.0+
`step`	No	No	5.0+	6.0+	10.6+
`list`	10.0+	4.0+	No	20.0+	9.0+
`novalidate`	10.0+	4.0+	No	10.0+	9.0+

4 Video e audio con HTML 5

Tra le novità che HTML5 offre rispetto alle versioni precedenti, vi è la possibilità di usufruire dei contenuti multimediali utilizzando due elementi: `<video>` e `<audio>`. L'introduzione della possibilità di riprodurre filmati o tracce audio in un browser senza l'ausilio di plug-in di terze parti, come Flash, rappresenta di per sé una grossa novità.

Questi due elementi sin dall'inizio sono quelli che hanno destato maggior interesse riguardo a questa nuova tecnologia, non solo nei confronti di piccoli gruppi di programmatori web ma di grandi aziende come Apple, YouTube e Microsoft.

In precedenza l'inserimento di questo tipo di contenuti all'interno di una pagina HTML richiedeva, il più delle volte, l'utilizzo di applicativi esterni (e non standard) come Adobe Flash. L'alternativa a Flash era sottostare a condizioni molto restrittive per quanto riguarda la multimedialità, condizioni oggi inidonee a soddisfare le specifiche minime richieste da molti dei servizi del Web 2.0 (si pensi, ad esempio, a YouTube).

Inizialmente furono sollevate alcune perplessità riguardanti il video e l'audio di HTML 5 perché inadatti a proteggere adeguatamente i contenuti e ancora impreparati all'integrazione di inserzioni pubblicitarie. Ben presto però arrivarono le librerie JavaScript pronte a risolvere questi problemi. Inoltre tali librerie mirano ad eliminare problemi di compatibilità con vecchi browser che non supportano tali elementi. Sono state create anche altre librerie, come *PopCorn.js,* con il fine di migliorare l'accessibilità dei contenuti fornendo ulteriori interfacce agli utenti.

Elemento <video>

Nella sua forma più semplice il tag video ricorda il tag img: `<video src="video.ogv">`. Sarebbe fin troppo bello se fosse sufficiente questa dichiarazione per integrare un video in un sito web. Purtroppo, ancora una volta, i principali browser non supportano le stesse funzionalità. In questo caso l'argomento riguarda i codec per la riproduzione video, quindi prima della presentazione delle funzionalità dell'elemento, qui sotto è riportata una tabella con i formati compatibili dei vari browser.

Formato					
`MP4`	9.0+	NO	3.2+	4.0+	No
`WebM`	No	4.0+	No	6.0+	10.6+
`Ogg`	No	3.5+	No	4.0+	10.5+

- MP4 = documenti MPEG 4 con codec video H264 e codec audio AAC
- WebM = documeti WebM con codec video VP8 e codec audio Vorbis
- Ogg = documenti Ogg con codec video Theora e codec audio Vorbis

Per quanto riguarda Safari è necessario installare *Quick Time* per visualizzare i video.

Elemento <source>

Per riprodurre un video in tutti i principali browser bisognerà associare all'elemento più formati dello stesso video. Questo si può fare grazie all'elemento `<source>` ripetuto più volte all'interno dei tag `<video></video>`. Gli attributi di <source> sono: `src` il cui valore corrisponde al percorso associato al video e `type` che permette di specificare la codifica video.

Attributo controls

Aggiungendo questo attributo booleano vengono visualizzati i classici bottoni per la riproduzione video con il fine di poter tenere sotto controllo l'esecuzione. Google Chrome, Safari e FireFox tramite l'utilizzo di un pulsante permettono di visualizzare il video a schermo intero.

Attributo poster

L'immagine visualizzata prima dell'esecuzione del video corrisponde al primo fotogramma. Se si desidera cambiarla è sufficiente impostare il valore dell'attributo `poster` all'interno del tag `<video>` con il percorso dell'immagine.

Attributo preload

Questo attributo può assumere tre valori:

- `none`: indica allo user-agent di non effettuare nessun scaricamento preventivo del contenuto del file video, il buffer inizierà quindi ad essere riempito alla prima pressione del tasto *play*.
- `metadata`: prima dell'esecuzione vengono recuperate solamente le informazioni essenziali come durata, dimensione, ecc...
- `auto`: impostando questo valore si presuppone che l'utente voglia prendere visione del video, quindi il browser caricherà preventivamente il video.

Attributo autoplay

Come si può intuire questo attributo booleano consente l'esecuzione automatica del video.

Attributo loop

Anche questo è un attributo booleano che se abilitato, il video inizia una nuova esecuzione una volta che è giunto al termine.

Attributo muted

Attributo booleano che disabilita preventivamente l'audio, ma permette comunque all'utente di riabilitarlo (a patto che siano presenti i controlli).

Attributi width e height

Permettono di definire la larghezza e l'altezza del video. Se non impostati, saranno stabiliti dal browser in funzione dell'immagine associata all'attributo `poster`, se presente, oppure in funzione del primo fotogramma del video.

Elemento <track>

Nessuno dei principali browser supporta questo elemento che contenuto all'interno dei tag `<video></video>` dovrebbe contenere elementi testuali che si possono combinare con il video (ad esempio i sottotitoli). Per il momento si può sostituire questa funzione con la libreria *Popcorn.js*.

Esempio video

```
<video controls autoplay>
        <source
        src="http://www.quirksmode.org/html5/videos/big_buck_bunny.mp4"
        type="video/mp4" >
        <source
        src="http://www.quirksmode.org/html5/videos/big_buck_bunny.ogv"
        type="video/ogg" >
        <source
        src="http://www.quirksmode.org/html5/videos/big_buck_bunny.webm"
        type="video/webm">
</video>
```

Elemento <audio>

Come per il tag `<video>` anche il tag `<audio>`, se tutti i browser supportassero almeno un formato comune, si potrebbe integrare in modo semplice ed elegante: `<audio src="audio.oga" controls>`. Ecco una tabella che riporta i vari formati audio supportati dai browser più popolari.

Formato					
MP3	9.0+	NO	4.0+	4.0+	NO
Wav	No	4.0+	4.0+	6.0+	10.6+
Ogg	No	3.5+	NO	4.0+	10.5+

Elemento <source>

Anche il tag `<audio>`, per risolvere le problematiche relative all'ascolto della traccia audio su tutti i browser, può contenere questo elemento permettendo di associare ad una traccia diversi formati.

Attributi

Gli attributi dell'elemento audio sono: `src`, `preload`, `autoplay`, `loop` e `controls` e le loro funzionalità sono analoghe a quelle dell'elemento `<video>` . Bisogna notare che in assenza dell'attributo `controls` i controlli audio non sono visibili di conseguenza tutto l'elemento sarà invisibile.

Esempio audio

```
<audio controls autoplay>
        <source src="http://alexandre.alapetite.fr/doc-alex/html5-
        audio/DTMF1000ms.mp3" type="audio/mp3">
        <source src="http://alexandre.alapetite.fr/doc-alex/html5-
        audio/DTMF1000ms.oga" type="audio/ogg"
        onerror="alert('Impossibile riprodurre il file audio')" >
</audio>
```

JavaScript

L'elemento `<audio>` dispone anche di un proprio costruttore JavaScript, ecco la sintassi:

Audio con javascript

```
var audio = new Audio("file_musicale.mp3");
```

In questo modo è possibile utilizzare musiche ed effetti audio senza necessariamente creare l'elemento HTML né ricorrere a `display:none` o similari.

5 Canvas

Come ormai è risaputo il tag `<canvas>` è stata una delle più grandi rivoluzioni di HTML 5. Questo nuovo elemento si può interpretare come una tela sulla quale disegnare con il "pennello" Canvas 2D API, ovvero un insieme di funzioni JavaScript che permettono un rendering dinamico di immagini bitmap. Canvas permette applicazioni e sviluppi un tempo assolutamente impensabili, se non utilizzando Flash o risorse esterne.

In questa sezione saranno presentati solamente i metodi legati al contesto di disegno 'Canvas 2d', ma esiste già una versione sperimentale di contesto *Canvas 3d* che sottende un set di API completamente diverso, basato sulle *OpenGL ES 2.0* e detto *WebGL*.

Elemento <canvas>

Gli unici attributi specifici di questo elemento sono `width` e `height` che permettono di impostare le dimensioni della "tela". Se non impostati i valori sono 300 pixel di larghezza e 150 pixel di altezza. Gli elementi all'interno dei tag `<canvas></canvas>` sono visualizzati solamente se il tag non è supportato dal browser, in questo caso si potrebbe inserire un'immagine come alternativa all'elemento.

Prima di cominciare a disegnare è necessario ottenere un riferimento all'oggetto che rappresenta l'elemento con il metodo JavaScript `getElementById()` e verificare che il browser lo supporti. Con l'utilizzo della libreria *Modernizr* si può testare il supporto.

Test supporto elemento <canvas>

```
<script>
        var canvas = document.getElementById("tela")
        if(Modernizr.canvas){
                //si scrive sul canvas
        }
</script>
```

Dopo aver testato il supporto, utilizzando il metodo `getContext()`, si ottiene un riferimento al contesto bidimensionale nell'ambito del quale operare. L'unico argomento che, a oggi, accetta è `2d`. L'oggetto che si ottiene si può immaginare come un piano cartesiano dove l'origine è l'angolo in alto a sinistra e il valore dell'asse delle ascisse e delle ordinate aumenta man mano che ci si allontana dall'origine spostandosi rispettivamente verso destra e verso il basso.

Metodi e proprietà

Le tabelle di seguito riportate contengono una breve descrizione dei metodi e delle proprietà dell'oggetto rappresentante l'elemento `<canvas>`. I metodi possono accettare uno o più argomenti.

Colori, stili ed ombre

Proprietà	*Descrizione*
`fillStyle`	Imposta o restituisce il colore, il gradiente o il modello del disegno.
`strokeStyle`	Imposta o restituisce il colore, il gradiente, o il modello dei tratti.
`shadowColor`	Imposta o restituisce il colore usato per le ombre.
`shadowBlur`	Imposta o restituisce il livello di sfocatura per le ombre.
`shadowOffsetX`	Imposta o restituisce la distanza orizzontale dell'ombra dalla forma.
`shadowOffsetY`	Imposta o restituisce la distanza verticale dell'ombra dalla forma.

Metodo	*Descrizione*
`createLinearGradient()`	Crea un gradiente lineare (da usare sul contenuto del Canvas).
`createPattern()`	Ripete un elemento specificato nella direzione specificata.
`createRadialGradient()`	Crea un gradiente radiale/circolare (da usare sul contenuto del Canvas).
`addColorStop()`	Specifica i colori e le posizioni di arresto in un oggetto gradiente.

Stili delle linee

Proprietà	*Descrizione*
`lineCap`	Imposta o restituisce lo stile delle punte di una linea.
`lineJoin`	Imposta o restituisce il tipo di angolo creato, quando due linee si incontrano.
`lineWidth`	Imposta o restituisce la larghezza di una linea.
`miterLimit`	Imposta o restituisce l'angolazione oltre la quale la punta della giunzione ad angolo di un segmento viene troncata.

Rettangoli

Metodo	*Descrizione*
`rect()`	Creazione di un rettangolo.
`fillRect()`	Disegna un rettangolo "pieno".
`strokeRect()`	Disegna un rettangolo (senza riempimento).
`clearRect()`	Cancella i pixel specificati all'interno di un rettangolo dato.

Percorsi

Metodo	*Descrizione*
`fill()`	Riempie il disegno corrente (percorso).
`stroke()`	Disegna il percorso che è stato definito.
`beginPath()`	Inizia un percorso, o reimposta il percorso corrente.
`moveTo()`	Consente di spostare il percorso in un punto specificato del Canvas.
`closePath()`	Crea un nuovo percorso dalla fine del punto del percorso corrente.
`lineTo()`	Aggiunge un nuovo punto e crea una linea da quel punto all'ultimo punto specificato del Canvas.
`clip()`	Taglia una regione di qualsiasi forma e dimensione dalla tela originale.
`quadraticCurveTo()`	Crea una curva quadratica di Bézier.
`bezierCurveTo()`	Crea una curva di Bézier cubica.
`arc()`	Crea un arco/curva (utilizzato per creare cerchi, o parti di cerchi).
`arcTo()`	Crea un arco/curva tra due tangenti.
`isPointInPath()`	Restituisce `true` se il punto specificato è nel percorso corrente, altrimenti `false`.

Trasformazioni

Metodo	*Descrizione*
`scale()`	Applica una scala al disegno corrente.
`rotate()`	Ruota il disegno corrente.
`translate()`	Trasla il disegno corrente.
`transform()`	Sostituisce la matrice di trasformazione corrente del disegno.
`setTransform()`	Reimposta la matrice di trasformazione alla matrice identità.

Testi

Proprietà	*Descrizione*
`font`	Imposta o restituisce le proprietà del font.
`textAlign`	Imposta o restituisce l'allineamento del testo.
`textBaseline`	Imposta o restituisce la baseline del testo.

Metodo	*Descrizione*
`fillText()`	Disegna il testo creato nel Canvas.
`strokeText()`	Disegna il testo creato nel Canvas (senza riempimento).
`measureText()`	Restituisce un oggetto che contiene la larghezza del testo specificato.

Immagini

Metodo	*Descrizione*
`drawImage()`	Disegna un'immagine, tela, o un video nel Canvas.

Manipolazione dei pixel

Proprietà	*Descrizione*
`width`	Restituisce la larghezza di un oggetto `ImageData`.
`Height`	Restituisce l'altezza di un oggetto `ImageData`.
`Data`	Restituisce un oggetto che contiene i dati di un oggetto `ImageData` specificato.

Metodo	*Descrizione*
`createImageData()`	Crea un nuovo oggetto vuoto `ImageData`.
`getImageData()`	Restituisce un oggetto `ImageData` contenente le informazioni dei pixel del rettangolo specificato sul Canvas.
`putImageData()`	Inserisce i pixel di un oggetto `ImageData` nel Canvas.

Composizione

Property	*Descrizione*
`globalAlpha`	Imposta o restituisce il valore di trasparenza del disegno.
`globalCompositeOperation`	Imposta o restituisce il modo in cui le immagini vengono sovrapposte.

Un semplice esempio pratico

L'esempio di seguito riportato è un semplice script che disegna, all'interno dell'elemento `<canvas>`, un rettangolo colorato con ombreggiatura.

Esempio <canvas>

```
<canvas id="tela" width="150" height="150">
        Il browser che sta utilizzando non supporta l'elemento canvas
</canvas>
<script type="text/javascript">
        var canvas = document.getElementById("tela")
        contesto = canvas.getContext("2d");

        contesto.shadowColor = '#003300'
        contesto.shadowOffsetX = 10;
        contesto.shadowOffsetY = 10;
        contesto.shadowBlur =20;
        contesto.fillStyle= "rgba(50, 255, 255, 0.2)"
        contesto.rect(10,20,100,120);
        contesto.stroke();
        contesto.fill();
</script>
```

Dopo aver ottenuto il riferimento al contesto vengono impostate le proprietà che insistono sul *drawing state* del Canvas relative all'ombreggiatura che successivamente verrà applicata a tutti gli oggetti che seguono.

Il colore predefinito è il nero, quindi per modificare il colore si dovrà impostare la proprietà `fillStyle`. Il colore si può applicare tramite le stesse proprietà già note per i fogli di stile. Nell'esempio è stato utilizzato `rgba(50, 255, 255, 0.2)` dove I parametri sono rispettivamente: tonalità rosso [0-255], tonalità verde [0-255], tonalità blu [0-255] e trasparenza [0-1].

Il metodo `rect(10,20,100,120)` disegna un rettangolo posizionato nel punto (10, 20) di lunghezza 100px e altezza 120px.

Il metodo `stroke()` esegue l'operazione di scrittura all'interno dell'elemento `<canvas>` dei tratti preventivamente impostati. Il metodo `fill()` esegue un'operazione analoga con la differenza che il primo agisce sui bordi mentre il secondo agisce nell'area. Nell'esempio, `stroke()` disegna il bordo del rettangolo (se non lo si elimina esplicitamente è presente) e dell'ombreggiatura, `fill()` applica il colore all'interno del rettangolo e dell'ombreggiatura.

Altre tecnologie

Come già detto HTML5, in questo caso l'elemento `<canvas>`, non sostituirà Flash cosi facilmente. Inoltre esiste anche SVG per quanto riguarda il rendering di immagini.

Attualmente riassumendo con una breve lista si potrebbe dire che:

SVG

- Non completamente supportato da tutti i browser.
- Adatto alla manipolazione di immagini vettoriali.
- Semplice gestione degli eventi.
- Meno efficiente del Canvas.
- Possibilità di disegno e gestione di immagini complesse grazie anche al supporto di prodotti di larga diffusione (Adobe Illustrator CS5, ad esempio).

CANVAS

- Supportato dai principali browser.
- Permette di manipolare le immagini bitmap in maniera approfondita.
- Difficile gestione degli eventi.
- Più performante di SVG.
- Spesso richiede librerie JavaScript apposite.

FLASH

- Creazione di animazioni anche senza codice.
- Ridotto supporto e performance su browser mobile.
- Semplice gestione degli eventi.
- Più performante di Canvas e SVG su computer desktop.
- Tecnologia proprietaria.

Tabella di compatibilità

					
`<canvas>`	9.0+	2.0+	3.1+	4.0+	9.0+

6 Microdata

La specifica dei microdati HTML 5 è un modo per assegnare etichette ai contenuti al fine di descrivere un tipo specifico d'informazioni (ad esempio recensioni, informazioni su persone o eventi). Ogni tipo d'informazione descrive uno specifico tipo di elemento, come una persona, un evento o una recensione. Ad esempio, un evento ha proprietà quali il luogo, l'ora d'inizio, il nome e la categoria.

L'obbiettivo è quello di esaltare le proprietà semantiche per dare la possibilità a programmi come *crawler* dei motori di ricerca o *screen reader* di comprendere il significato del testo. Queste informazioni sono accessibili da questi programmi e rimangono (per ora) invisibili per l'utente.

Sintassi

Marcare i dati pubblicati con il formato microdata è semplice, è sufficiente applicare i seguenti attributi ai tag HTML in modo tale da definire gli oggetti semantici.

Itemscope

Questo è un attributo globale che contrassegna un insieme di proprietà logicamente legate tra loro. Assegnando questo attributo si precisa che all'interno dell'elemento che lo contiene sono presenti informazioni dell'oggetto che si vuole descrivere.

Itemprop

Definisce il nome della proprietà associata all'oggetto che si vuole descrivere. L'attributo può essere definito, ad esmpio, in un tag `<span></span>`, in questo caso il valore della proprietà sarà il contenuto dell'elemento ma se `itemprop` è definito, ad esempio, in un tag `<img>` il valore della proprietà va ricercato all'interno dell'elemento. Nel caso dell'immagine, il valore della proprietà definita sarà il valore dell'attributo `src`.

Itemtype

Questo attributo permette la "tipizzazione" degli elementi mircodata. Serve principalmente a distinguere una proprietà con un determinato nome associata a un determinato oggetto, da una proprietà con lo stesso nome ma associata a un oggetto da descrivere differente. In sintesi il valore di `itemtype` delinea il contesto delle proprietà definendo così un *vocabolario*. Conviene utilizzare vocabolari che siano ampiamente diffusi e condivisi in modo tale da favorire la condivisione delle informazioni. Il valore di `itemtype` può quindi essere un collegamento ipertestuale che fa riferimento ad un vocabolario condiviso in rete.

Esempio microdata

```
<div itemscope itemtype="http://data-vocabulary.org/Person">
        Il mio nome è
        <span itemprop="name">Federico Toffano</span>
        e il mio nickname in alcuni siti web è
        <span itemprop="nickname">fede89</span>.
        Questo è il mio account twitter:
        <a href="http://www.example.com"
        itemprop="url">http://www.example.com</a>
        Vivo a Treviso in Italia e sono uno
        <span itemprop="role">studente</span>.
</div>
```

Rich snippet

I microdati stanno alla base dei *rich snippet* di Google ovvero le informazioni aggiuntive visualizzate dopo una ricerca. Google ha rilevato come i collegamenti che offrono una sintesi ben strutturata delle proprie informazioni abbiano sperimentato un incremento di traffico rilevante. Ecco un esempio:

Figura 1: rich snippet su Google Chrome

Per definire i microdata in modo tale che siano interpretabili dai *rich snippet* visitare la pagina web nella quale sono presenti i vocabolari di Google: http://www.data-vocabulary.org/ .

I vocabolari più popolari, supportati da Google, sono:

- *Breadcrumbs*: serve a rappresentare un insieme di collegamenti che possano aiutare un utente a comprendere e navigare all'interno del sito.
- *Businesses and organizations*: definisce una società, un negozio e più in generale un luogo.
- *People*: definisce una persona.
- *Address*: definisce un indirizzo.
- *Events*: definisce un evento (con informazioni come il titolo, la data e il luogo).
- *Product*: definisce un prodotto.
- *Offer*: definisce un'offerta.
- *Offer-aggregate*: definisce un aggregato di offerte (con prezzo minimo, prezzo massimo, etc).
- *Recipes*: definisce una ricetta.
- *Review*: definisce una singola recensione.
- *Review-aggregate*: definisce una recensione aggregata.
- *Rating*: definisce una valutazione.

7 Applicazioni web offline

HTML5 introduce le*offline API* che permettono la memorizzazione in locale di documenti per la visualizzazione di pagine web anche in assenza di una connessione internet. Le informazioni riguardanti i file necessari alla navigazione offline sono definite all'interno di un file dedicato con estensione *manifest*.

Questa novità, ad esempio, potrebbe risultare molto utile per la navigazione web tramite dispositivi mobili in aree in cui non c'è una buona copertura od è completamente assente. Esistono già alcune applicazioni che sfruttano questa nuova tecnologia, un esempio è *Google Gears*.

File manifest

Innanzitutto è necessario aggiungere l'attributo `manifest` all'elemento `<html>` segnalando così allo user agent l'esistenza del file con le specifiche per una consultazione offline. Il valore di questo attributo deve corrispondere al percorso del file contenente l'elenco dei documenti per i quali si richiede la memorizzazione. Il file in questione dovrà essere *MIME type* `text/cache-manifest`, quindi si dovrà inoltre impostare il server in modo tale da servire il file con il *MIME type* corretto.

Esempio file manifest

```
CACHE MANIFEST

#commento

CACHE:
/pagina.html
/css/cssPagina.css

NETWORK:
/inviaDocumento.aspx

FALLBACK:
news/* /offline.html
/contatti.aspx /contatti.html
```

Sezione CACHE

Se vengono inseriti indirizzi di alcune risorse senza dichiarare una sezione, implicitamente viene applicata la sezione `CACHE`. Questa sezione rappresenta le risorse che si devono rendere disponibili nel caso in cui non si riesca a stabilire una connessione con il server.

Sezione NETWORK

Questa sezione, come `CACHE`, contiene un'unica risorsa per riga. In questo caso però gli indirizzi indicati si riferiscono a risorse per le quali il reperimento richiede una connessione internet attiva.

Sezione FALLBACK

In questa sezione, diversamente dalle precedenti, per ogni riga sono indicate due risorse distinte. La prima indica quella disponibile nel caso di possibile accesso alla rete. La seconda rappresenta invece l'alternativa di cui disporre in caso di offline. Nell'esempio la prima riga di questa sezione indica che tutto il contenuto della cartella `news`, nel caso di mancato accesso alla rete, sarà sostituito con la pagina `offline.html`.

Commenti

Il file *manifest* include anche la possibilità di inserire dei commenti, questo si può fare solamente inserendo all'inizio della riga da commentare il simbolo '#'. Questa limitazione è imposta per evitare ambiguità che potrebbero sorgere nel caso in cui il commento sia inserito in una riga contenente un indirizzo web.

Funzionamento

Per capire come sono gestite le applicazioni web offline si possono descrivere 4 casi distinti durante la visitazione di un sito web che offre tale servizio.

Prima visita

Se il browser visita una pagina web nella quale sarà rilevato un riferimento a un file *manifest* per la prima volta, significa che l'utente non conserva una copia in locale di tale file. A questo punto il browser scarica il file, lo interpreta e richiede al server tutte le risorse indicate al suo interno che saranno necessarie in caso di mancato accesso alla rete. Le fasi che compongono il recupero delle informazioni associate a questo caso (prima visita) sono demarcate dai seguenti eventi:

- *checking:* analisi del file *manifest*.
- *downloading:* scaricamento di tutti i file.
- *progress:* per ogni risorsa da scaricare viene scatenato tale evento.
- *cached:* operazioni concluse.

Visita successiva senza modificazione del file *manifest*

Il browser rileva che è già stato scaricato un file *manifest* associato a questo sito web quindi, invece di rieseguire le operazioni corrispondenti alla prima visita, si limita al controllo del file locale e del file server per la rilevazione di eventuali modifiche. In questo caso non sarà individuata alcuna modifica quindi non seguiranno altre fasi. Gli eventi di questo processo sono:

- *checking:* analisi del file *manifest*.
- *noupdate:* evento che comunica l'assenza di differenze tra i due file *manifest*.

Visita successiva con modifica del file *manifest*

Analogamente al caso precedente sarà rilevata la presenza del file manifest. In questo caso però durante l'analisi saranno rilevate modifiche, quindi il file sarà sostituito con quello presente sul server. In seguito saranno scaricate in locale tutte le risorse elencate. Gli eventi di questo scenario sono i seguenti:

- *cecking:* analisi del file *manifest*.
- *downloading:* scaricamento dei file causato dalla differenza dei due file.
- *progress:* evento scatenato per ogni risorsa da scaricare in locale.
- *updateready:* segnala che tutte le risorse sono state nuovamente scaricate in locale.

Mancanza di risorse definite all'interno del file *manifest*

Durante lo scaricamento dei file necessari potrebbe accadere che una o più risorse non siano raggiungibili. In questo caso sarà scatenato un evento con il fine di comunicare un errore durante l'esecuzione. Gli eventi associati a questo caso saranno:

- *checking:* analisi del file *manifest*.
- *downloading:* notifica scaricamento delle risorse.
- *progress:* evento scatenato per ogni risorsa da scaricare in locale.
- *error:* evento scatenato a causa dell'impossibilità di scaricare una determinata risorsa in locale.

Tabella di compatibilità

					
Applicazioni web offline	10.0+	3.5+	4.0+	4.0+	10.6+

8 Web Storage

Le WebStorage API permettono il salvataggio locale d'informazioni concernenti la navigazione web. Nascono per risolvere due principali problematiche relative ai cookie: Lo spazio per la memorizzazione, che passa dai restrittivi 4KB a 5MB, e l'impossibilità di avere due sessioni distinte in un browser relative allo stesso dominio. Prima del Web Storage di HTML 5 ci furono altri tentativi per la risoluzione di queste due problematiche senza però riscontrare successo perché basate su tecnologie proprietarie (come *userData* di Microsoft) o perché necessitavano l'installazione di componenti aggiuntivi (*flash script*).

Fortunatamente tutti i browser supportano questo nuovo membro della specifica, di conseguenza molto probabilmente, in poco tempo, il Web Storage prenderà il sopravvento sui cookie.

sessionStorage e localStorage

I problemi legati ai cookie sono stati risolti con l'introduzione di due nuovi oggetti.

sessionStorage

Consente di avere un meccanismo di persistenza dei dati distinto per ogni sessione di navigazione in ogni finestra, o scheda, del browser eliminando i dati alla chiusura. Usando `sessionStorage` ad esempio sarebbe quindi possibile coordinare l'apertura contemporanea di due distinti account GMail sullo stesso browser. Questo oggetto è da utilizzare nel caso in cui le informazioni da memorizzare sono di carattere temporaneo, quindi per processi di breve durata.

localStorage

Mantiene il comportamento dei cookie essendo comune a tutte le finestre del browser che condividono lo stesso dominio. I dati persistono anche dopo la chiusura del browser. Permette ad esempio di aggiungere articoli nel carrello e di poter completare l'acquisto nei giorni successivi. L'utilizzo ideale quindi riguarda applicazioni in cui un processo può essere interrotto per poi essere ripreso anche a distanza di giorni.

Entrambi sono inoltre stati studiati per ospitare molti più dati, fino a 5MB, e a differenza dei cookie i dati non sono scambiati sistematicamente tra client e server, generando richieste meno corpose.

Le specifiche

Entrambi gli oggetti, `sessionStorage` e `localStorage`, implementano l'interfaccia `Storage` che raggruppa un insieme di metodi comuni ai due oggetti. La memorizzazione è consentita solo per valori testuali e avviene tramite una coppia chiave valore. I metodi per accedervi sono: `setItem()` per valorizzare la chiave e `getItem()` per ottenerne il valore. In realtà, in formato stringa, si possono anche memorizzare valori numerici interi o decimali che in seguito, con l'ausilio dei metodi `parseInt()` o `parseFloat()`, si possono riottenere in formato numerico.

JSON

Nel caso in cui si richieda la memorizzazione di un oggetto, si può utilizzare `JSON` (*JavaScript Object Notation*), un semplice formato utilizzato per lo scambio di dati. La funzione

`JSON.stringfy()` permette di ottenere una rappresentazione stringa di un oggetto mentre la sua funzione inversa è rappresentata da `JSON.parse()` che permette di riottenere il formato originale.

Esempio sessionStorage

```
<script type="text/javascript">
	var oggetto = {
		"nome" : "Federico",
		"cognome": "Toffano",
		"email" : "email@gmail.com"
	};
	stringaOggetto = JSON.stringify(oggetto);
	sessionStorage.setItem("credenziali", stringaOggetto);
	stringaOggettoRipristinato =
	sessionStorage.getItem("credenziali");
	var oggettoRipristinato = JSON.parse(stringaOggetto);
	alert(oggettoRipristinato["email"]);
</script>
```

Nell'esempio si poteva procedere in modo analogo con l'oggetto `localStorage`.

Per quanto riguarda la memorizzazione di variabili stringhe i metodi `sessionStorage.setItem("chiave","valore")` e `temp=sessionStorage.getItem("chiave")` si possono sostituire rispettivamente con `sessionStorage.chiave="valore"` e `temp=sessionStorage.chiave`.

Intercettare modifiche alle variabili

Nel caso in cui fosse necessario rilevare eventuali modifiche alle variabili memorizzate, si può intercettare l'evento `storage`, notificato ogni qualvolta i dati sono modificati. Questo evento mette a disposizione una serie di proprietà, accessibili solamente in lettura, contenenti alcune informazioni relative alla modifica effettuata:

- `key`*:* la chiave dell'oggetto modificato.
- `oldValue`*:* il valore associato alla chiave prima della modifica.
- `newValue`*:* il valore associato alla chiave dopo la modifica.
- `url`*:* l'indirizzo del documento in cui ha avuto luogo la modifica.
- `storageArea`*:* riferimento all'oggetto, che rappresenta i dati, nel quale è stata rilevata la modifica.

Esempio evento storage

```
<script type="text/javascript">
	if(window.addEventListener){
		window.addEventListener("storage", modificaDati, false);
	}
	else if(window.attachEvent){ //per Internet Explorer 6,7 e 8
		Window.attachEvent("onstorage", modificadati);
	}

	function modificaDati(event){
		var dataStorage = event || window.event;
		alert("chiave: " + dataStorage.key + " – valore:" +
		dataStorage.newValue);
	}
</script>
```

Privacy

I cookie in passato sono stati visti come una minaccia alla privacy, molti utenti hanno deciso di disabilitarli con la conseguente riduzione dei servizi online. Senza cookie ad esempio non si riesce ad utilizzare Gmail, Twitter e molte altre applicazioni web. Il Web Storage amplificando le potenzialità dei cookie rende ancora più sensibile il tema della privacy. Ad esempio se un ente in grado di distribuire i suoi contenuti su più siti assegnasse un identificativo univoco salvato nell'area di storage locale di un utente, potrebbe essere in grado di tracciare i suoi movimenti nel web. Così facendo sarebbe più semplice focalizzare i suoi interessi in modo tale da poter predisporre della pubblicità mirata.

Tabella di compatibilità

					
WebStorage	8.0+	2.0+	4.0+	4.0+	10.5+

9 Geolocalizzazione

Le *Geolocation API* in realtà non fanno parte della specifica HTML 5, anche se spesso sono associate a quest'ultima (persino articoli di stampa specialistica sbagliano). Gli strumenti di relativi alla geolocalizzazione sono oggetto di standardizzazione da parte del *Geolocation Working Group*.

Questa tecnologia è stata sviluppata in concomitanza con HTML 5 ed è anch'essa una componente innovativa della programmazione web. Molte applicazioni in rete hanno già cominciato a farne uso, soprattutto nel campo mobile.

Geolaction API

I principali metodi associati alla geoloacalizzazione sono `getCurrentPosition` e `watchPosition`, entrambi utili allo stesso scopo: ottenere la posizione corrente. La principale differenza tra i due è che il primo rileva la posizione un'unica volta mentre il secondo dopo la prima rilevazione, nel caso in cui la posizione abbia subito una variazione significativa, trasmette una notifica contenente le nuove informazioni. Inoltre `watchPosition` restituisce un identificativo numerico univoco che servirà a localizzare i successivi spostamenti del dispositivo, per terminare la lettura ripetuta delle coordinate è necessario invocare il metodo `clearWatch` che accetta come argomento tale identificativo. Entrambe le funzioni accettano tre argomenti: `successCallback`, `errorCallback` e `options`.

Parametro successCallback

Funzione da eseguire nel caso in cui il rilevamento della posizione è andato a buon fine. Alla funzione è passato come argomento un riferimento all'oggetto `Position`. Quest'ultimo ha due proprietà: la data e l'ora del rilevamento della posizione contenute all'interno della proprietà `timestamp` e l'oggetto `Coords` che permette l'accesso alle seguenti proprietà:

- `longitude`: informazioni di longitudine.
- `latitude`: informazioni di latitudine.
- `altitude`: informazioni relative all'altitudine. Se il dispositivo su cui è installato il browser non riesce a rilevare l'altitudine, il parametro sarà `null`.
- `accuracy`: indica un valore in metri che rappresenta l'approssimazione dei valori di longitudine e latitudine.
- `altitudeAccuracy`: indica un valore in metri che rappresenta l'approssimazione del valore di altitudine. Se il valore di *altitude* è null allora lo sarà anche quello di *altitudeAccuracy*.
- `heading`: indica la direzione espressa in gradi. Il valore corrisponde a una rotazione oraria rispetto al nord. Se il dispositivo è stazionario, il valore sarà `NaN`.
- `speed`: velocità espressa in metri al secondo. Se la velocità è zero *heading* sarà `NaN`.

Parametro errorCallback

Parametro facoltativo che rappresenta la funzione da eseguire nel caso in cui il rilevamento della posizione sia fallito. Alla funzione è passato come argomento un riferimento all'oggetto `PositionError` contenente le due proprietà `message` e `code` utili a comprendere la natura dell'errore. In particolare la proprietà `code` può assumere i seguenti valori interi:

- *1*: PERMISSION_DEINED, l'utente non ha dato il consenso relativo alla rilevazione della posizione.
- *2*: POSITION_UNAVAILABLE, Il tracciamento è fallito a causa di un errore relativo al fornitore dei dati.
- *3*: TIMEOUT, il tempo prestabilito entro il quale il tracciamento avrebbe dovuto avere luogo è stato superato.

Parametro options

è un oggetto di tipo `PositionOptions` le cui tre proprietà `enableHighAccuracy`, `timeout` e `maximumAge` permettono di intervenire nel modo in cui avviene il tracciamento:

- `enableHighAccuracy`: è una proprietà booleana che se impostata a `true` viene richiesto di determinare la posizione nel modo più accurato possibile. Le conseguenze però sono un maggior tempo di elaborazione e un aumento del consumo energetico.
- `timeout`: è la soglia di tempo prefissata entro la quale si desidera una risposta. Se la soglia viene superata l'esecuzione fallisce. Il valore è espresso in millisecondi e nel caso in cui non venga impostata una soglia il valore è pari a zero.
- `maximumAge`: è un intervallo di tempo espresso in millisecondi che, nel caso di acquisizioni della posizione ripetute, stabilisce quanto tempo deve trascorrere per l'aggiornamento dei valori.

Reverse geocoding

Con il termine *Reverse geocoding* s'intende una metodologia che permette, tramite le informazioni di latitudine e longitudine, di ricavare altre informazioni semplici da comprendere come il nome di una via o di una località. Nel web un modo semplice per farlo potrebbe essere quello di sfruttare le potenzialità di Google Maps. Un esempio si trova alla fine di questa sezione.

Privacy

Parlando di geolocalizzazione sicuramente il tema della privacy è un argomento da non sottovalutare. Fortunatamente se un sito web ospita tale tecnologia, l'utilizzo è consentito solo col previo consenso dell'utente.

Figura 1: Richiesta geolocalizzazione Google Chrome

Inoltre, dopo aver dato il consenso ad un determinato sito web, il browser deve permettere l'annullamento del consenso.

Esempio geolocalizzazione con l'utilizzo di Google Maps

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false">
//importo libreria Google Maps
</script>
<div id="mappa" style="height:200px; width:300px; float:left;
overflow:hidden;"></div>
<div id="dati"></div>
<script type="text/javascript">
        //recupero i dati relativi alla posizione
        navigator.geolocation.getCurrentPosition(geoCompletata,
        geoFallita,
        {'enableHighAccuracy':true,'timeout':10000,'maximumAge':0} );
        //geolocalizzazione completata con successo
        function geoCompletata(posizione){
                //leggo le informazioni del metodo getCurrentPosition
                var geoDati =
                "latitude: "+ posizione.coords.latitude +"<br>\n"+
                "longitude: "+ posizione.coords.longitude +"<br>\n"+
                "accuracy: "+ posizione.coords.accuracy +"<br>\n"+
                "altitude: "+ posizione.coords.altitude +"<br>\n"+
                "altitudeAccuracy: "+ posizione.coords.altitudeAccuracy
                +"<br>\n"+
                "heading: "+ posizione.coords.heading +"<br>\n"+
                "speed: "+ posizione.coords.speed +"";
                //stampo a video le infromazioni
                document.getElementById("dati").innerHTML = geoDati;
                //riferimento all'oggetto da utilizzare all'interno della
                //mappa contenente latitudine e longitudine
                var coordinate = new
                google.maps.LatLng(posizione.coords.latitude,
                posizione.coords.longitude);
                // opzioni relative alla mappa da visualizzare
                var opzioni = {zoom: 15, mapTypeId:
                google.maps.MapTypeId.ROADMAP, center: coordinate};
                //creazione della mappa con le relative opzioni
                var mappa = new
                google.maps.Map(document.getElementById("mappa"),
                opzioni);
                //creazione del segnaposto all'intenro della mappa
                segnaposto = new google.maps.Marker({map:mappa,
                draggable:true, animation: google.maps.Animation.DROP,
                position: coordinate});
        }

        //geolocalizzazione fallita: gestione degli errori
        function geoFallita(errore){
                if(errore.code == 1) {
                        alert("L'utente non ha autorizzato la
                geolocalizzazione");
                }
                else if(errore.code == 2) {
                        alert("Posizione non disponibile");
                }
                else if(errore.code == 3){
                        alert("Timeout");
                }
                else{
                        alert("ERRORE:" + errore.message);
                }
        }
</script>
```

Tabella di compatibilità

					
Geolocation	9.0+	3.5+	5.0+	5.0+	10.6+

10 Le altre API DI HTML 5

Indexed Database API

Queste API danno la possibilità di gestire un database memorizzato all'interno del browser dell'utente. La struttura è basata su oggetti di tipo `Object Store`, paragonabili ad array associativi ordinati, dove ogni coppia chiave-valore rappresenta un oggetto. Come per i classici database è possibile eseguire operazioni d'inserimento, modifica, eliminazione e ricerca.

Web Workers API

I Web Workers permettono l'esecuzione di codice JavaScript in parallelo, in modo tale da non interferire con le prestazioni relative alla visualizzazione della pagina web. Sono file JavaScript che possono essere paragonati a dei thread con i quali la pagina web può interagire tramite alcuni metodi dedicati. La specifica prevede che un WebWorker possa essere un oggetto JavaScript di due tipi: `Worker` e `SharedWorker`. La principale differenza sta nel fatto che il secondo può comunicare in modo bidirezionale con più sessioni a differenza del primo che sarà associato ad un'unica sessione.

Web socket API

Le WebSockets API permettono di stabilire una connessione bidirezionale tra browser e server tramite l'oggetto JavaScript `WebSocket`. Lo scambio delle informazioni avviene tramite il metodo `send` per l'invio e l'evento `onmessage` per la ricezione. Le informazioni sono scambiate tramite il protocollo `The` *WebSocket Protocol*, creato per queste API.

Drag and drop

Le Drag and Drop API permettono di dotare le applicazioni web del *Drag and Drop* senza dover utilizzare librerie o framework JavaScript esterni e tra di loro incompatibili. In breve, è possibile rendere sorgente del drag (operazione di trascinamento) o destinatario del drop(operazione di rilascio) qualunque elemento HTML, sia all'interno della finestra dove sono stati definiti tali elementi, sia con altre finestre del browser stesso. I principali elementi coinvolti in questo processo sono:

- `draggable`: attributo che permette di rendere trascinabile un elemento HTML.
- `ondrop`: evento che viene invocato al rilascio di un elemento sopra un area di rilascio.

Tabella di compatibilità

API HTML 5					
`Indexed Database`	10.0+	4.0+	No	11.0+	No
`Web Workers`	10.0+	3.5+	4.0+	4.0+	10.6+
`Web Socket`	10.0+	4.0+	5.0+	4.0+	11.0+
`Drag and Drop`	5.5+	3.5+	3.1+	4.0+	12.0+

11 Nuovi selettori e pseudo-classi

I CSS (Cascading Style Sheet) forniscono un supporto al linguaggio HTML per quanto riguarda la visualizzazione delle pagine. Nonostante i bachi e le svariate interpretazioni, i vari browser sono riusciti, nel tempo, ad ottenere un ruolo fondamentale per quanto riguarda la formattazione dei documenti web.

La nuova specifica CSS 3, rispetto alle versioni precedenti, adotta un approccio modulare. Ciascun modulo quindi compre una determinata area, di conseguenza il ciclo di vita e l'avanzamento è diverso per ognuno di essi.

Per quanto riguarda la retrocompatibilità fortunatamente esistono alcune strategie di supporto alternativo in modo tale da poter trascurare le incompatibilità con vecchie versioni dei browser.

Selettori di attributo

I selettori di attributo permettono di identificare un attributo in funzione di una parte o dell'intero valore assegnato, associando un determinato stile all'elemento che lo contiene. Il problema relativo all'utilizzo di questo tipo di selettori riguarda le prestazioni: calcolare a quanti e quali elementi applicare la regola di stile rallenta il caricamento della pagina. Le tipologie di selettori di attributo sono le seguenti:

- *E[attribute]*
- *E[attribute=value]*
- *E[attribute~=value]*
- *E[attribute|=value]*
- *E[attribute^=value]*
- *E[attribute$=value]*
- *E[attribute*=value]*

In realtà i primi quattro facevano già parte della precedente specifica. Le novità della versione 3 dei CSS sono gli ultimi tre dell'elenco.

E[attribute]

La regola si applica all'elemento di tipo `E` contenente l'attributo `attribute`. Nell'esempio lo stile sarà applicato al primo paragrafo.

Esempio E[attribute] stile

```
p[id] { color: blue; }
```

Esempio E[attribute] applicazione

```
<p id="prova">Lorem ipsum</p>
<p>testo</p>
```

E[attribute=value]

La regola si applica all'elemento di tipo `E` contenente l'attributo `attribute` con valore `value`. Nell'esempio lo stile sarà applicato al primo collegamento ipertestuale.

Esempio E[attribute=value] stile

```
a[href="#"] { text-decoration: overline; }
```

Esempio E[attribute=value] applicazione

```
<a href="#"> Lorem ipsum </a>
<a href="http://www.prova.it"> Lorem ipsum </a>
```

E[attribute~=value]

La regola si applica all'elemento di tipo E contenente l'attributo attribute con valore composto da più parole, separate da spazi, tra le quali almeno una di esse è uguale a value. Nell'esempio lo stile sarà applicato al primo collegamento ipertestuale.

Esempio E[attribute~=value] stile

```
a[title~="applicato"] { text-decoration: line-through; }
```

Esempio E[attribute~=value] applicazione

```
<a title="stile applicato" href="#"> Lorem ipsum </a>
<a title="stile diverso" href="#> Lorem ipsum </a>
```

E[attribute|=value]

La regola si applica all'elemento di tipo `E` contenente l'attributo `attribute` con valore composto da più parole separate da un trattino tra le quali la prima è uguale a `value`. Nell'esempio lo stile sarà applicato al primo collegamento ipertestuale.

Esempio E[attribute|=value] stile

```
a[title|="lorem"] { text-decoration: underline; }
```

Esempio E[attribute|=value] applicazione

```
<a title="lorem-ipsum" href="#"> Lorem ipsum </a>
<a title="ipsum-lorem" href="#> Lorem ipsum </a>
```

E[attribute^=value]

Questo è il primo dei tre nuovi selettori di attributo. La regola si applica all'elemento di tipo `E` contenente l'attributo `attribute` con valore che comincia per `value`. Nell'esempio lo stile sarà applicato al primo collegamento ipertestuale.

Esempio E[attribute^=value] stile

```
img[title^="lorem"] { border:5px solid red; }
```

Esempio E[attribute^=value] applicazione

```
<img title="loremipsum" src="lorem.jpg"> Lorem ipsum </img>
<img title="lORemipsum" src="lorem.jpg"> Lorem ipsum </img>
```

E[attribute$=value]

Questo è il secondo dei tre nuovi selettori di attributo. La regola si applica all'elemento di tipo `E` contenente l'attributo `attribute` con valore che termina per `value`. Nell'esempio lo stile sarà applicato al primo collegamento ipertestuale.

Esempio E[attribute$=value] stile

```
img[title$="Ipsum"] { border:5px solid red; }
```

Esempio E[attribute$=value] applicazione

```
<img title="loremIpsum" src="lorem.jpg"> Lorem ipsum </img>
<img title="loremIpsumSit" src="lorem.jpg"> Lorem ipsum </img>
```

E[attribute*=value]

Questo è il terzo dei tre nuovi selettori di attributo. La regola si applica all'elemento di tipo `E` contenente l'attributo `attribute` con valore che contiene `value`. Nell'esempio lo stile sarà applicato al primo collegamento ipertestuale.

Esempio E[attribute=value] stile*

```
img[title^="Lorem"] { border:5px solid red; }
```

Esempio E[attribute=value] applicazione*

```
<img title="IpsumLoremIpsum" src="lorem.jpg"> Lorem ipsum </img>
<img title="Ipsum lorem Ipsum" src="lorem.jpg"> Lorem ipsum </img>
```

Pseudo-classi strutturali

Le pseudo-classi di struttura permettono di applicare un determinato stile a un elemento in funzione della posizione occupata nella struttura HTML della pagina web. La sintassi è analoga a quelle delle precedenti specifiche: una pseudo-classe si definisce assegnando simbolo ":" seguito dal nome della pseudo-classe.

Di seguito sono riportate tutte le pseudo-classi strutturali. Da notare che per un qualsiasi elemento contenente degli elementi figli, l'indice dei figli, contrariamente ai linguaggi di programmazione, inizia da 1 e non da 0.

E:first-child

Questa regola era già presente nella precedente versione della specifica e permette di applicare lo stile al solamene al primo nodo figlio dell'elemento `E`.

E:last-child

La regola permette di applicare lo stile all'ultimo nodo figlio dell'elemento `E`.

E:empty

La regola si applica all'elemento di tipo `E` con la condizione che quest'ultimo non abbia nodi figli. Quindi lo stile, ad esempio, non sarà applicato nemmeno ad un paragrafo se questo contiene del testo.

E:root

Identifica la radice della pagina. Per quanto riguarda i documenti HTML la radice è `<html>` quindi le seguenti due regole producono lo stesso effetto.

```
html { margin:3px; }
:root { margin:3px; }
```

E:only-child

La regola si applica all'elemento di tipo `E` con la condizione che quest'ultimo sia l'unico figlio di un elemento HTML padre.

E:only-of-type

La regola si applica all'elemento di tipo `E` con la condizione che quest'ultimo sia l'unico figlio, del tipo specificato di un elemento HTML padre.

Esempio E:only-of-type stile

```
a:only-of-type { color:red; }
```

Esempio E:only-of-type: campo di applicazione

```
<div>
        <p> Lorem ipsum </p>
        <a href="#"> Lorem ipsum </a>
</div>
```

Nell'esempio sopra riportato sarà applicato lo stile al collegamento ipertestuale poiché è l'unico all'interno dell'elemento `<div>`.

Esempio E:only-of-type: campo di esclusione

```
<div>
        <a href="#"> Lorem ipsum 1</a>
        <a href="#"> Lorem ipsum 2</a>
</div>
```

Nell'esempio sopra riportato non sarà applicato lo stile a nessun collegamento ipertestuale poiché sono presenti due elementi dello stesso tipo all'interno dell'elemento padre `<div>`.

E:first-of-type

La regola si applica al primo elemento di tipo `E` figlio di un elemento HTML padre.

E:last-of-type

La regola si applica all'ultimo elemento di tipo `E` figlio di un elemento HTML padre.

E:nth-child(n)

La regola si applica all'ennesimo figlio di un elemento HTML padre con la condizione che sia un elemento di tipo `E`. In alternativa questa pseudo-classe si può applicare agli elementi di tipo `E`, figli di un elemento HTML padre, con indice `an+b` dove `a` e `b` sono numeri interi.

Esempio E:nth-child(n) stile

```
tr:nth-child(2n) { background-color:red; }
tr:nth-child(2n-1) { background-color:yellow; }
```

Esempio E:nth-child(n): campo di applicazione

```
<table>
	<tr><th>Cognome</th><th>Nome</th><th>Eta</th></tr>
	<tr><td>Rossi</td><td>Luca</td><td>17</td></tr>
	<tr><td>Bianchi</td><td>Marco</td><td>23</td></tr>
	<tr><td>Ferrari</td><td>Giovanna</td><td>29</td></tr>
</table>
```

Questo esempio rappresenta una tabella con sfondi di riga alterni. In alternativa è stata introdotta una sintassi più semplice che permette di ottenere lo stesso effetto:

Esempio E:nth-child(n) stile alternativo

```
tr:nth-child(even) { background-color:red; }
tr:nth-child(odd) { background-color:yellow; }
```

E:nth-last-child(n)

Questa regola è analoga alla precedente con un'unica differenza: l'indice n dei figli di un elemento HTML corrisponde all'ennesimo figlio partendo dall'ultimo. Applicando le due seguenti righe CSS all'esempio precedente si ottiene la stessa tabella dell'esempio precedente con sfondi di riga opposti.

Esempio E:nth-last-child(n) stile

```
tr:nth-last-child(2n) { background-color:red; }
tr:nth-last-child(2n-1) { background-color:yellow; }
```

E:nth-of-type(n)

Questa regola è una variante di E:nth-child(n): si applica all'ennesimo elemento di tipo `E` figlio di un elemento HTML padre. La differenza è che gli indici sono assegnati solamente agli elementi di tipo `E` e non a tutti i figli dell'elemento HTML padre.

Esempio E:nth-of-type(n) stile

```
div a:nth-of-type(even) { color:red; }
div a:nth-of-type(odd) { color:blue; }
```

Esempio E:nth-child(n): campo di applicazione

```
<div>
	<a href="#"> Lorem ipsum </a>
	<p> Lorem ipsum </p>
	<a href="#"> Lorem ipsum </a>
</div>
```

Se all'esempio sopra riportato fosse stata applicata la pseudo classe `nth-child`, l'indice del secondo collegamento ipertestuale sarebbe stato 3 mentre con la pseudo classe `nth-of-type` vale 2.

E:nth-last-of-type(n)

Questa regola è analoga alla precedente con un'unica differenza: l'indice n dei figli di tipo E, di un elemento HTML, corrisponde all'ennesimo figlio partendo dall'ultimo.

Pseudo-classi per la validazione

Queste pseudo-classi permettono di definire gli stili per i vari stati degli elementi `E` nel contesto dell'interfaccia utente. Il principale utilizzo si riscontra nei web from 2.0, infatti, si

può notare un'analogia tra gli stati che possono assumere i nuovi *input type* e i nomi di queste nuove pseudo classi. L'elenco completo delle nuove pseudo-classi per la validazione è il seguente:

- *E:enabled*
- *E:disabled*
- *E:checked*
- *E:default*
- *E:valid*
- *E:invalid*
- *E:required*
- *E:optional*
- *E:in-range*
- *E:out-of-range*
- *E:read-only*
- *E:read-write*

Esempio pseudo-classi per la validazione: stile

```
input:valid { background: green }
input:invalid { background: red }
input:in-range { background:green }
input:out-of-range { background:red }
```

Esempio pseudo-classi per la validazione: campo di applicazione

```
<form name="intervista" method="post" action="/send">
	<fieldset>
		<legend>Intervista</legend>
		<p>
			<label> Come valuti questo sito? </label>
		<br>
			<input type="number" min="0" max="10" step="1"
			value="11">
		</p>
		<p>
			<label for="email">Email*</label>
			<br>
			<input id="email" type="email" name="email"
			placeholder="tua@mail.it" size="40" required>
		</p>
		<input type="submit" value="Inoltra">
	</fieldset>
</form>
```

Nell'esempio se I valori dei campi rispettano le regole assegnate, lo sfondo sarà verde, altrimenti sarà rosso.

Altre pseudo-classi

Oltre alle pseudo-classi strutturali e alle pseudo-classi per la validazione ne sono state introdotte altre:

E:target

Questa pseudo-classe permette di associare uno stile a un elemento `E` all'interno di una pagina, associato a un collegamento ipertestuale nella medesima pagina, nel momento in cui s'invoca l'evento *click* del collegamento.

Esempio E:target stile

```
div:target { border: 3px solid red }
```

Esempio E:target campo di applicazione

```
<a href="#id1"> Lorem ipsum 1</a>
<a href="#id2"> Lorem ipsum 2</a>
<a href="#id3"> Lorem ipsum 3</a>

<div id="id1">
       <p> Lorem Ipsum è un testo segnaposto utilizzato nel settore
       della tipografia e della stampa.</p>
</div>
<div id="id2">
       <p> Lorem Ipsum è considerato il testo segnaposto standard sin
       dal sedicesimo secolo, quando un anonimo tipografo prese una
       cassetta di caratteri e li assemblò per preparare un testo
       campione</p>
</div>
<div id="id3">
       <p> È sopravvissuto non solo a più di cinque secoli, ma anche al
       passaggio alla videoimpaginazione, pervenendoci sostanzialmente
       inalterato.</p>
</div>
```

Nell'esempio, invocando l'evento *click* del primo o del secondo o del terzo collegamento ipertestuale, si evidenzia con un bordo rosso il `div` associato al collegamento.

E:not

Questa è pseudo-classe di negazione ovvero lo stile è associato a tutti gli elementi di tipo E che non coincidono con il selettore contenuto all'interno delle parentesi seguite dal `:not`.

Esempio E:not stile

```
div:not(#id1) { border: 3px solid red }
```

Esempio E:not campo di applicazione

```
<div id="id1">
       <p> Al contrario di quanto si pensi, Lorem Ipsum non è
       semplicemente una sequenza casuale di caratteri. </p>
</div>
<div id="id2">
       <p> Risale ad un classico della letteratura latina del 45 AC,
       cosa che lo rende vecchio di 2000 anni. <p>
</div>
<div id="id3">
       <p> La prima riga del Lorem Ipsum, "Lorem ipsum dolor sit
       amet..", è tratta da un passaggio della sezione 1.10.32. </p>
</div>
```

Nell'esempio gli ultimi due `div` avranno il bordo rosso a differenza del primo.

Tabella di compatibilità

Nuove pseudo-classi strutturali					
`E:first-child`	7.0+	1.0+	1.0+	1.0+	7.0+
`E:root`	9.0+	1.0+	1.0+	1.0+	9.5+
`E:nth-child()`	9.0+	3.5+	3.1+	1.0+	9.5+
`E:nth-last-child()`	9.0+	3.5+	3.1+	1.0+	9.5+
`E:last-child`	9.0+	1.0+	3.1+	1.0+	9.5+
`E:only-child`	9.0+	1.5+	3.1+	1.0+	9.5+
`E:nth-of-type()`	9.0+	3.5+	3.1+	1.0+	9.0+
`E:nth-last-of-type()`	9.0+	3.5+	3.1+	1.0+	9.5+
`E:first-of-type`	9.0+	3.5+	3.1+	1.0+	9.5+
`E:last-of-type`	9.0+	3.5+	3.1+	1.0+	9.5+
`E:only-of-type()`	9.0+	3.5+	3.1+	1.0+	9.5+
`E:empty`	9.0+	1.5+	2.0+	1.0+	9.5+

Nuove pseudo-classi per la validazione					
`E:default`	No	3.5+	4.0+	2.0+	9.0+
`E:valid`	No	4.0+	4.0+	2.0+	9.0+
`E:invalid`	No	4.0+	4.0+	2.0+	9.0+
`E:in-range`	No	No	No	7.0+	9.0+
`E:out-of-range`	No	No	No	7.0+	9.0+
`E:read-only`	No	No	No	No	No
`E:read-write`	No	No	No	No	No

Altre pseudo-classi					
`E:target`	9.0+	1.0+	3.1+	1.0+	10.5+
`E:not`	9.0+	1.5+	1.0+	1.0+	9.5+

12 Web Fonts

Il modulo CSS Fonts dei CSS 3, tramite la direttiva `@font-face`, permette l'utilizzo di un qualsiasi font in modo semplice e veloce. Non è più necessario ricorrere a tecniche alternative, è sufficiente affidarsi a quanto fissato nelle specifiche del W3C a proposito dei cosiddetti *web fonts*. In realtà la direttiva `@font-face` non è del tutto una novità, nella specifica CSS 2 tale funzionalità era già prevista, ma in seguito eliminata nella specifica CSS 2.1. Già con la versione 4.0 di Internet Explorer (rilasciato 1º ottobre 1997 per Windows) i *web fonts*, caricati con `@font-face`, erano supportati. Attualmente tutti i browser più popolari supportano tale modulo quindi non c'è motivo di esitare davanti alla possibilità di sfruttare questa tecnica.

La classica assegnazione di un font a uno stile CSS è la seguente:

Proprietà font-family

```
p:{font-family: Arial, Verdana;}
```

Impostando la proprietà `font-family` in questo modo, il browser cercherà di utilizzare il primo font. Nel caso in cui non sia disponibile, proverà con il secondo.

@font-face

La regola `@font-face` permette di assegnare e rendere interpretabile qualsiasi tipo di font indipendentemente dalla macchina che visiterà il sito web. Questo è possibile perché sarà presente un collegamento ad un file contenente tutte le direttive necessarie all'interpretazione del font.

Esempio sintassi @font-face

```
@font-face{
        font-family: 'MioFont';
        src: url('MioFont.eot');
        src:
                local('☺'),
                url('MioFont.woff') format('woff'),
                url('MioFont.ttf') format('truetype');
}
```

Nell'esempio è riportata la tecnica corss-browser di Paul Irish (uno dei due creatori di Modernizr) che permette una corretta interpretazione dei font sui vari browser:

- `font-family: 'MioFont'`: Nome arbitrario da assegnare al font, verrà riutilizzato per poterlo assegnare ai vari stili CSS.
- `url('fonts/MioFont.eot')`: Questo è l'unico formato interpretabile da Internet Explorer per le versioni precedenti alla 9. Gli utenti che utilizzano uno di questi browser scaricheranno implicitamente il file indicato, mentre gli altri browser ignoreranno questa direttiva.
- `local('☺')`: Indica al browser di cercare in locale un font con il nome indicato. La faccina serve per escludere che qualcuno possa avere un font installato con lo stesso nome. Se il font fosse individuato, ci sarebbe il rischio che non corrisponda a quello desiderato.

- `url('MioFont.woff') format('woff')`: indica al browser di scaricare il file relativo al font nel formato `woff`. Questo formato è interpretabile da Safari, Google Chrome, Internet Explorer dalla 9 e Firefox dalla 3.6.
- `url('MioFont.ttf') format('truetype')` : indica al browser di scaricare il file relativo al font nel formato `ttf`. Questo formato è interpretabile da Firefox 3.5, versioni vecchie di Safari e Google Chrome, Opera dalla 10, iPhone, iPod e iPad.

Font Squirrel

http://www.fontsquirrel.com è un sito web che offre la possibilità di scaricare un pacchetto zip contenente tutti i file nei vari formati di un font. Oltre ad esserci numerosi font disponibili, offre un servizio, chiamato *@font-face Generator*, che a partire da un font caricato dall'utente in un determinato formato, esegue la conversione del font nei vari formati per poterli poi applicare tramite CSS.

Cufón

Nel caso in cui `@font-face` non sia supportato si può utilizzare un generatore di caratteri chiamato `Cufón`. Il sito web http://cufon.shoqolate.com/generate/ offre un servizio che genera un file JavaScript a partire da un font caricato dall'utente. Per rendere il font interpretabile durante la visitazione del sito web è necessario includere i seguenti file:

- *cufon-yui.js*: Il file che permette la generazione dei caratteri.
- *MioFont*.js: Il file generato dal servizio online di `Cufón`.
- *jQuery.js*: Libreria di supporto a *cufon-yui.js.*

Tabella di compatibilità

Font					
`TTF`	9.0+	3.5+	3.1+	4.0+	10.0+
`OTF`	9.0+	3.5+	3.1+	4.0+	10.0+
`WOFF`	9.0+	3.6+	5.1+	5.0+	11.1+
`SVG`	No	No	3.2+	4.0+	9.0+
`EOT`	6.0+	No	No	No	No

13 Effetti tipografici

Essendo che la definizione della specifica CSS3 è ancora in corso, presso i principali browser è possibile aggiungere dei prefissi alle proprietà in modo tale da permettere l'implementazione delle nuove funzionalità per cui il processo di standardizzazione sia ancora incompleto:

- `-webkit`: Safari e Google Chrome
- `-o`: Opera
- `-moz`: FireFox
- `-ms`: Internet Expolorer

Saranno utilizzati solo in questa fase di transizione e nel prossimo futuro saranno completamente obsoleti. Anche se alcune proprietà sono supportate dalle versioni più recenti dei browser omettendo i prefissi proprietari, per mantenere la compatibilità con alcune vecchie versioni potrebbe essere necessario applicarli comunque.

Word-wrap

Questa proprietà nasce per risolvere il problema relativo a parole troppo lunghe in spazi troppo corti. Più precisamente permette di riportare una parola su più righe se questa si estende al di fuori dell'area in cui altrimenti sarebbe stata confinata. In questo modo si evitano svorapposizioni di elementi o la comparsa di barre di scorrimento indesiderate. I valori che `word-wrap` può assumere sono due:

- `normal`: è il valore predefinito, applica quindi il comportamento classico.
- `break-word`: è il valore che permette l'applicazione del `word-wrap` all'elemento che contiene lo stile.

Esempio word-wrap

```
#contenitore{
        width: 200px;
        background: red;
        word-wrap: break-word;
}
```

Text-shadow

Uno degli effetti tipografici più diffusi oggi è applicabile anche agli stili CSS: grazie alla proprietà `text-shadow`, definita nella specifica CSS 3, si ha la possibilità di creare un testo ombreggiato.

Inizialmente, come per `@font-face`, la proprietà era già prevista nella specifica CSS 2, ma solamente Safari la supportava. In seguito, con la definizione della specifica CSS 2.1, tale proprietà fu rimossa. Ora, ricomparsa con la specifica CSS 3, è supportata da tutti i principali browser. Per risolvere l'incompatibilità con le versioni di Internet Explorer inferiori alla 10, si può ricorrere ai filtri *Shadow e Drop Shadow* sviluppati da Microsoft.

La proprietà `text-shadow` permette quindi di ottenere effetti di ombreggiatura nel testo delle pagine web senza dover ricorrere ad immagini apposite risparmiando così tempo di sviluppo e banda. I valori da assegnare a questa proprietà sono quattro:

- Il primo definisce lo scostamento lungo l'asse x del testo ombreggiato.
- Il secondo definisce lo scostamento lungo l'asse y del testo ombreggiato.
- Il terzo esprime la sfumatura del testo ombreggiato tramite un raggio di sfumatura espresso in pixel. Impostando tale valore a zero si otterrà un'ombra netta senza sfuocatura.
- Il quarto definisce il colore del testo ombreggiato.

Esempio ombreggiatura

```
p{
        text-shadow: 10px 10px 10px #C0E1FF;
        font-size:50px;
}
```

Modificando i parametri si possono ottenere molti effetti, anche diversi dall'obreggiatura. Inoltre si possono applicare più obreggiature contemporaneamente separate da virgole aumentando così gli effetti ottenibili.

Esempio ombreggiatura Multipla

```
p{
        text-shadow: 10px 10px 10px #C0E1FF, 10px 20px 10px #C0E1FF;
        font-size:50px;
}
```

Nell'esempio viene applicata la stessa ombra per due volte ma con scostamento lungo l'asse y diverso.

Text-overflow

Nel caso in cui ci sia del testo, posto all'interno di un contenitore non sufficientemente grande per contenerlo, si può applicare la proprietà `text-overflow` con valore `ellipsis` in modo tale da troncare il testo aggiungendo i tre punti di sospensione.

Esempio text-overflow stile

```
#contenitore{
        width: 200px;
        background: red;
        overflow:hidden;
        white-space:nowrap;
        text-overflow: ellipsis;
}
```

Esempio text-overflow applicazione

```
<div id="contenitore">testo con punti di sospensione</div>
```

Nello stile dell'esempio, per visualizzare l'effetto di `text-overflow`, è stato necessario aggiungere alcune proprietà:

- `width`: imposta la dimensione lungo l'asse x del contenitore
- `overflow hidden`: nasconde il contenuto che fuoriesce dal contenitore
- `white-space nowrap`: impone al browser di non andare a capo se sono presenti spazi bianchi e la riga è stata riempita.

Box-sizing

Con la specifica CSS 2.1, quando s'impostano le dimensioni di un contenitore, la regola del *box model* prevede che la larghezza e l'altezza sono definite rispettivamente tramite i valori delle proprietà `width` e `height`. Questi due valori non considerano le dimensioni impostate per il padding, il bordo e il margine. Questa regola può causare perdite di tempo e poca flessibilità nel caso in cui si debbano eseguire modifiche strutturali. Questo perché per conoscere l'esatta dimensione occupata dal contenitore creato è necessario sommare i valori dei vari attributi.

La specifica CSS 3 introduce la proprietà `box-sizing` per evitare questi inconvenienti. Tale proprietà può assumere due valori:

- `content-box`: il valore delle proprietà `width` e `height` seguono la regola del box model di CSS 2.1
- `border-box`: il valore della proprietà `width` e `height` include le dimensioni di padding, bordo e margine.

Attualmente, per quanto riguarda le versioni più recenti dei browser, solo FireFox necessita del prefisso proprietario.

Testo in colonne

Con il modulo CSS 3 chiamato *Multi column layout* è stata introdotta la possibilità di disporre del testo contenuto in un singolo contenitore su più colonne. Le principali proprietà applicabili, attualmente supportate da tutti i browser, sono le seguenti:

- `column-width`: valore che indica la larghezza minima delle colonne.
- `column-count`: valore intero che indica il numero di colonne.
- `column-gap`: valore che indica la distanza tra le colonne.
- `column-rule`: questa proprietà è composta da tre valori. Il primo indica la larghezza della riga, il secondo lo stile (gli stili sono analoghi a quelli di `border-style` della specifica CSS 2.1) e l'ultimo definisce il colore.

Esempio testo in colonne stile

```
#contenitore{
        width: 500px;
        column-count:2;
        column-gap:20px;
        column-rule: 1px dotted black;
}
```

Attualmente FireFox, Google Chrome e Safari necessitano del prefisso proprietario per questa proprietà.

Display: box

Applicando questo nuovo valore alla proprietà `display` di un contenitore si può gestire la disposizione degli elementi al suo interno in modo molto rapido. Questa innovazione è stata introdotta per offrire un'alternativa più flessibile rispetto i classici `float:left` e `float:right`. Il contenitore dovrà quindi essere associato ad uno stile contenente la proprietà `display:box`. Per la gestione della disposizione degli elementi al suo interno si utilizzeranno le seguenti proprietà:

- `box-orient`: se il valore assegnato è `vertical` gli elementi saranno disposti in verticale altrimenti se il valore assegnato è `horiziontal` gli elementi saranno disposti in orizzontale.
- `box-direction`: se impostato a `reverse` inverte l'ordine in cui verranno visualizzati gli elementi altrimenti `normal` mantiene l'ordine preimpostato.
- `box-align`: il suo valore determina l'allineamento sull'asse verticale se gli elementi sono orientati in senso orizzontale (`box-orient: horizontal;`) e l'allineamento sull'asse orizzontale se gli elementi sono orientati in senso verticale (`box-orient: vertical;`). I valori disponibili sono: `start`, `end`, `center`, `baseline` e `stretch`.
- `box-pack`: gestisce l'allineamento in funzione della disposizione degli elementi impostata con la proprietà `box-orient`. Inoltre gestisce lo spazio vuoto non occupato dagli elementi all'interno dell'elemento contenitore. I valori disponibili sono `start`, `end`, `center` e `justify`.

Esistono inoltre altre due proprietà che a differenza delle precedenti si possono applicare agli elementi contenuti nel contenitore:

- `box-ordinal-group`: permette di stabilire l'ordine in cui gli elementi saranno visualizzati assegnando un intero che indica l'indice dell'elemento.
- `box-flex`: questa proprietà permette di definire lo spazio occupato dall'elemento in funzione del numero di elementi presenti nel contenitore e dalla dimensione del contenitore. Ad esempio assegnando a tutti gli elementi il valore 1, lo spazio occupato sarà uguale per ogni elemento. Oppure assegnando ad un elemento il valore 2 questo avrà una dimensione doppia rispetto agli altri elementi con valore impostato a 1.

Attualmente tutti i browser che supportano questo componente necessitano dei prefissi proprietari.

Tabella di compatibilità

Font					
`text-shadow`	10.0+	3.5+	4.0+	4.0+	9.5+
`word-wrap`	5.5+	15.0+	3.1+	4.0+	No
`text-overflow`	6.0+	7.0+	3.1+	4.0+	9.0+
`overflow-x`	9.0+	3.5+	4.0+	4.0+	9.5+
`overflow-y`	9.0+	3.5+	4.0+	4.0+	9.5+
`box-sizing`	8.0+	2.0+	3.1+	4.0+	9.5+
`testo in colonne`	10.0+	2.0+	3.1+	4.0+	11.10+
`display:box`	10.0+	3.0+	3.0+	4.0+	No

14 Bordi e sfondi

CSS Backgrounds and Borders Module Level 3 è un modulo che si occupa della definizione delle proprietà relative ai bordi e agli sfondi.

Border-radius

Questa proprietà permette la realizzazione di angoli arrotondati in modo semplice e intuitivo. In passato la soluzione per ottenere tal effetto prevedeva l'inserimento di quattro immagini negli angoli di una tabella e, di conseguenza, altrettante richieste HTTP addizionali. Ora impostando il valore di `border-radius` si definisce il raggio della sezione di cerchio necessaria a riprodurre l'effetto desiderato, un valore nullo definisce uno spigolo. I valori di questa proprietà possono essere uno, due, tre o quattro:

- Un valore: ogni angolo è rappresentato utilizzando lo stesso raggio.
- Due valori: il primo rappresenta il raggio relativo all'angolo in alto a sinistra e quello relativo all'angolo in basso a destra.
- Tre valori: il primo valore rappresenta il raggio relativo all'angolo in alto a sinistra, il secondo quello relativo all'angolo in alto a destra e in basso a sinistra ed infine il terzo valore è associato all'angolo in basso a destra.
- Quattro valori: rappresentano rispettivamente l'angolo in alto a sinistra, l'angolo in alto a destra, l'angolo in basso a destra e l'angolo in basso a sinistra.

In alternativa si può impostare il raggio di un singolo angolo tramite l'utilizzo delle seguenti proprietà:

- `border-top-left-radius`
- `border-top-right-radius`
- `border-bottom-right-radius`
- `border-bottom-left-radius`

Inoltre è possibile definire sia il raggio orizzontale sia quello verticale separando i due valori dal simbolo "/". L'angolo che si ottiene è rappresento da una sezione di elisse.

Esmpio border-radius css

```
#contenitore{
        width:200px;
        height:200px;
        border : 2px solid red;
        padding:20px;
        border-radius: 50px 25px 25px 50px / 50px 50px 25px 25px;
}
```

Border-image

Prima dell'introduzione dei CSS 3 al bordo si poteva applicare uno stile solamente tra quelli definiti dalla specifica precedente tramite la proprietà `border-style`. Ora con la proprietà `border-image` è possibile utilizzare un'immagine come bordo di un elemento. La proprietà è suddivisa in tre parti:

- *Prima parte*: indica il percorso dell'immagine da utilizzare.

- *Seconda parte*: supponendo che l'immagine caricata sia quella in figura 1, il valore assegnato la suddivide in 9 sezioni come in figura 2. Questo valore può essere espresso senza unità di misura (sottintendendo il pixel) oppure in percentuale. Rappresenta il lato dei quattro quadrati che identificano gli angoli. In realtà applicare un unico valore a questa parte è una semplificazione. Si potrebbero definire quattro valori: il primo rappresenta l'altezza della prima linea orizzontale a partire dal bordo superiore, il secondo rappresenta la larghezza della seconda linea verticale a partire dal bordo destro, il terzo rappresenta l'altezza della seconda riga orizzontale a partire dal bordo inferiore ed infine il quarto rappresenta la larghezza della prima linea verticale a partire dal bordo sinistro. La finalità di questa griglia è di identificare gli angoli dell'immagine che saranno utilizzati come angoli del bordo.
- *Terza parte*: se il valore è unico, definisce il comportamento dell'immagine nelle sezioni 2, 4, 5, 6 e 8 della griglia altrimenti si possono inserire due valori: il primo è relativo alle sezioni 2, 5 e 8 in senso verticale ed il secondo alle sezioni 4, 5 e 6 in senso orizzontale. I valori disponibili sono due: `round`, indica che l'immagine di quelle sezioni va ripetuta, o `stretch` che estende l'immagine lungo tutta la sezione.

Figura 1 – Immagine bordo

Figura 2 – griglia

In alternativa si possono definire le proprietà singolarmente:

- `border-image-source`
- `border-image-slice`
- `border-image-repeat`

Esempio border-image css

```
#contenitore{
        width:200px;
        height:200px;
        border : 2px solid red;
        padding:20px;
        border-image: url('immagine.png') 20 25 20 25 round stretch;
}
```

Nell'esempio è stato definito anche il classico `border` che in caso d'incompatibilità di `border-image` con il browser sarà comunque visualizzato un bordo. FireFox e Opera necessitano ancora dei prefissi proprietari, mentre Internet Explorer non supporta ancora tale proprietà.

Box-shadow

Questa proprietà permette di aggiungere un'ombreggiatura agli elementi contenitori. Tale proprietà accetta quattro argomenti:

- *Scostamento orizzontale*: valore espresso in pixel che indica lo scostamento dell'ombreggiatura lungo l'asse delle ascisse.
- *Scostamento verticale*: valore espresso in pixel che indica lo scostamento dell'ombreggiatura lungo l'asse delle ordinate.

- *Raggio di sfuocatura*: con valore alto relativo a questo parametro si ottiene una sfuocatura dell'ombreggiatura maggiore. Impostando un valore nullo si ottiene un'ombreggiatura netta.
- *Colore*: imposta il colore dell'ombreggiatura nel formato RGB.

Esempio box-shadow stile

```
#contenitore{
        width:200px;
        height:200px;
        border: 1px solid black;
        box-shadow: 10px 10px 2px red;
}
```

La specifica suggerisce di utilizzare questa proprietà anche per evidenziare delle parole all'interno di un testo con un riquadro ombreggiato.

Opacity

La proprietà `opacity` permette di definire il grado di trasparenza di un elemento. Il suo valore è compreso tra 0 (elemento invisibile) e 1 (trasparenza assente).

Esempio opacity stile

```
#contenitore{
        width:200px;
        height:200px;
        opacity:0.5;
        filter: alpha(opacity=50);
}
```

L'ultima riga dello stile relativo all'esempio è necessaria se si vuole estendere la compatibilità con Internet Explorer per le versioni precedenti alla 9.

Background-size

L'impossibilità di modificare le dimensioni dello sfondo era una limitazione non trascurabile, basti pensare che se l'immagine destinata allo sfondo, non avesse soddisfatto le esigenze richieste, l'unica soluzione fosse il ridimensionamento dell'immagine tramite qualche applicazione. La proprietà `background-size` di CSS 3 risolve questa problematica dando la possibilità di assegnare due valori: il primo rappresenta la dimensione lungo l'asse delle ascisse mentre il secondo la dimensione lungo l'asse delle ordinate.

In alternativa `background-size` accetta un argomento che può assumere due valori:

- `cover`: l'immagine viene ridimensionata con il fine di ricoprire interamente l'elemento a cui viene applicata; se il rapporto tra asse x e asse y dell'immagine e dell'elemento al quale verrà applicato lo sfondo sono differenti, verranno tagliate alcune parti dell'immagine.
- `contain`: l'immagine viene ridimensionata per adattarsi all'area dell'elemento cui viene applicata; a differenza di cover `questo` valore assicura che l'immagine sarà visualizzata completamente, con lo svantaggio che alcune aree dell'elemento potrebbero restare scoperte.

Background-origin

La proprietà `background-origin` serve a specificare il posizionamento dell'immagine di sfondo per mezzo di uno dei tre valori ammissibili:

- `padding-box`: il vertice in alto a sinistra dell'immagine coincide con l'angolo esterno in alto a sinistra del padding. Questa è il valore predefinito.
- `border-box`: il vertice in alto a sinistra dell'immagine coincide con l'angolo esterno in alto a sinistra del bordo;
- `content-box`: il vertice in alto a sinistra dell'immagine coincide con l'angolo interno in alto a sinistra del padding.

`background-origin` non può essere applicata se il valore della proprietà `background-attachment` corrisponde a `fixed`.

Background-clip

Il valore della proprietà `background-clip` specifica la sezione di un elemento su cui si estenderà lo sfondo:

- `border-box`: lo sfondo potrà estendersi fino a comprendere l'area del bordo. Questo è il valore predefinito.
- `padding-box`: lo sfondo potrà estendersi fino a comprendere l'area del padding escludendo l'area del bordo.
- `content-box`: lo sfondo potrà estendersi solo all'interno dell'elemento escludendo quindi bordo e padding.

Sfondi multipli

La specifica CSS 2 prevede la possibilità di assegnare un'unica immagine allo sfondo. Con CSS 3, invece, questa limitazione è superata. I nomi delle proprietà restano invariati ma ora i valori possono essere molteplici:

- `backgorund-image`: il valore può essere composto da più indirizzi separati da una virgola.
- `background-position`: i valori di questa proprietà, separati da una virgola, sono relativi alle immagini indicate nella proprietà `background-image` e definiscono la loro posizione
- `background-repeat`: l'assegnazione dei valori è analoga a quella di `background-position`. I valori ammessi sono gli stessi della specifica CSS 2 e definiscono la modalità di ripetizione dell'immagine.

Esempio sfondo multiplo stile

```
#contenitore{
	background-image: url(immagine1.png), url(immagine2.png);
	background-position: top left, 20% 80%;
	background-repeat: no-repeat, no-repeat;
}
```

Gradienti come immagini di sfondo

Con la specifica CSS 3, alla proprietà `background-image`, oltre a supportare il classico indirizzo url associato ad un'immagine, è possibile applicare un gradiente lineare o radiale.

Il gradiente lineare si applica tramite la funzione `linear-gradient()` che accetta i seguenti parametri separati da virgole:

- *Punto di partenza del gradiente*: questo valore è esprimibile tramite una delle parole chiave di posizione scelta tra `top`, `top left`, `top right`, `bottom`, `bottom left`, `bottom right`, `left` e `right`.
- *Colori di stop e offset*: Il colore di stop, ovvero la tonalità di colore prima di passare alla successiva tonalità, dev'essere espresso in forma esadecimale mentre l'offset, che specifica quale punto della linea del gradiente il colore di stop dovrà comparire, dev'essere espresso in percentuale. Si devono impostare almeno due colori di stop e offset separati da virgole, con la condizione che l'offset per ogni valore aggiunto sia crescente.

Esempio gradiente lineare css

```
#contenitore{
        width:200px;
        height:200px;
        background-image: -ms-linear-gradient(top right, #FFFFFF 0%,
        #AACFEF 100%);
        background-image: -moz-linear-gradient(top right, #FFFFFF 0%,
        #AACFEF 100%);
        background-image: -o-linear-gradient(top right, #FFFFFF 0%,
        #AACFEF 100%);
        background-image: -webkit-linear-gradient(top right, #FFFFFF 0%,
        #AACFEF 100%);
        background-image: linear-gradient(top right, #FFFFFF 0%, #AACFEF
        100%);
}
```

Il gradiente radiale si applica tramite la funzione `radial-gradient()` che accetta i seguenti parametri separati da virgole:

- *Punto di origine del gradiente*: questo valore è esprimibile tramite una delle parole chiave di posizione scelta tra `top`, `bottom`, `center`, `middle`, `left` e `right`.
- *Forma del gradiente e misura*: la forma può essere circolare (`circle`) oppure ellittica (`ellipse`, che è il valore preimpostato). La misura è relativa dimensioni del box su cui è applicato il gradiente, può assumere i valori `contain`, `farthest-side` o `closest-corner`.
- *Colori di stop e offset*: La sintassi è analoga a quella del gradiente lineare.

Esempio gradiente radiale css

```
#contenitore{
        width:200px;
        height:200px;
        background-image: -ms-radial-gradient(center, circle farthest-
        corner, #FFFFFF 0%, #00A3EF 100%);
        background-image: -moz-radial-gradient(center, circle farthest-
        corner,
        #FFFFFF 0%, #00A3EF 100%);
        background-image: -o-radial-gradient(center, circle farthest-
        corner,
        #FFFFFF 0%, #00A3EF 100%);
        background-image: -webkit-radial-gradient(center, circle
        farthest-corner,
        #FFFFFF 0%, #00A3EF 100%);
        background-image: radial-gradient(center, circle farthest-
        corner,
        #FFFFFF 0%, #00A3EF 100%);
}
```

Per le più recenti versioni di Internet Explorer, FireFox e Opera I prefissi proprietari non sono necessari.

Tabella di compatibilità

Bordi e sfondi					
border-radius	9.0+	4.0+	5.0+	4.0+	10.5+
border-image	No	3.5+	3.1+	4.0+	10.5+
box-shadow	9.0+	3.5+	3.1+	4.0+	10.5+
opacity	5.5+	2.0+	3.1+	4.0+	9.0+
sfondi multipli	9.0+	3.6+	3.1+	4.0+	10.5+
background-size	9.0+	3.6+	3.1+	4.0+	10.0+
background-origin	9.0+	3.6+	3.1+	4.0+	10.0+
background-clip	9.0+	3.6+	3.1+	4.0+	10.0+
Gradienti	10.0+	3.6+	4.0+	4.0+	11.10+

15 Trasformazioni

Le trasformazioni sono state definite nel *modulo CSS 2D Transforms*. Tale modulo mette a disposizione alcuni metodi per quanto riguarda la manipolazione 2D degli elementi contenuti nella pagina. La proprietà `transform`, applicabile a qualsiasi elemento, accetta come argomento una o più funzioni di trasformazione, mentre la proprietà `transform-origin` definisce il punto di origine da cui avviene la trasformazione. In data odierna le trasformazioni non necessitano i prefissi proprietari solo per le versioni più recenti di Internet Explorer, Opera e FireFox.

Rotate()

Questa funzione ruota in senso orario l'oggetto al quale è applicata. L'argomento di questo metodo rappresenta la rotazione in gradi; valori che non sono compresi tra 0 e 360 sono comunque consentiti e saranno implicitamente convertiti in valori appartenenti a questo intervallo. Valori negativi implicano una rotazione in senso antiorario.

Esempio rotate() css

```
#contenitore{
        width:200px;
        border: 1px solid blue;
        background: #c0e1ff;
        -webkit-transform: rotate(20deg);
        -moz-transform: rotate(20deg);
        -ms-transform: rotate(20deg);
        -o-transform: rotate(20deg);
        transform: rotate(20deg);
}
```

Scale(), scaleX(), scaleY()

La funzione `scale()` modifica le dimensioni dell'elemento a cui viene applicata. Se l'argomento è un unico numero, quest'ultimo rappresenta la scala da applicare sia lungo l'asse delle ascisse sia lungo l'asse delle ordinate. Due argomenti invece rappresentano rispettivamente la variazione della larghezza e della lunghezza. Altrimenti, se si vuole intervenire lungo un unico asse, bisogna applicare la scala tramite i metodi `scaleX()` e `scaleY()` che accettano come argomento un unico parametro. I parametri sono privi di unità di misura e l'esecuzione della funzione con argomento pari a 1 restituisce l'elemento invariato.

Esempio scale() css

```
#contenitore{
        width:200px;
        border: 1px solid blue;
        background: #c0e1ff;
        margin:50px;
        -webkit-transform: scale(0.8, 1.4);
        -moz-transform: scale(0.8, 1.4);
        -ms-transform: scale(0.8, 1.4);
        -o-transform: scale(0.8, 1.4);
        transform: scale(0.8, 1.4);
}
```

Skew(), skewX e skewY()

La funzione `skew()` distorce gli assi dell'elemento a cui viene applicata. Come per la funzione `scale()`, se l'argomento è unico la distorsione si applica in ugual modo lungo entrami gli assi, mentre se gli argomenti sono due la distorsione viene eseguita rispettivamente sull'asse delle ascisse e delle ordinate. Per intervenire lungo un unico asse si dovrà utilizzare `skewX()` e `skewY()`. Gli argomenti sono espressi in gradi e sono ammissibili anche valori negativi.

Esempio skew() css

```
#contenitore{
        width:200px;
        border: 1px solid blue;
        background: #c0e1ff;
        margin:50px;
        transform: skew(5deg, -20deg);
        -ms-transform: skew(5deg, -20deg);
        -moz-transform: skew(5deg, -20deg);
        -webkit-transform: skew(5deg, -20deg);
        -o-transform: skew(5deg, -20deg);
}
```

Translate(), translateX(), translateY()

Queste funzioni determinano una traslazione espressa in pixel lungo gli assi. La funzione `translate()` può essere eseguita con un parametro, applicando la traslazione lungo l'asse x, o con due parametri, specificando le traslazioni da applicare ad entrambi gli assi. Nel caso in cui si volesse traslare solamente l'asse y la funzione da usare è `translateY()`. La funzione `translateX()`, analoga a `translate()` invocata con un parametro, interviene solamente lungo l'asse delle ascisse. Parametri positivi indicano traslazioni verso destra o verso il basso.

Esempio translate() css

```
#contenitore{
        width:200px;
        border: 1px solid blue;
        background: #c0e1ff;
        transform: translate(10px, -10px);
        -ms-transform: translate(10px, -10px);
        -moz-transform: translate(10px, -10px);
        -webkit-transform: translate(10px, -10px);
        -o-transform: translate(10px, -10px);
}
```

Trasformazioni multiple

La proprietà `transform` permette di applicare in sequenza una serie di trasformazioni in modo tale da ottenere un risultato dato dalla somma degli effetti di ogni singola trasformazione.

Esempio trasformazioni multiple css

```
#contenitore{
        width:200px;
        border: 1px solid blue;
        background: #c0e1ff;
        transform: rotate(40deg);
        -webkit-transform: rotate(20deg) scale(0.8, 1.4)
        skew(5deg, -20deg);
        -moz-transform: rotate(20deg) scale(0.8, 1.4)skew(5deg, -20deg);
        -ms-transform: rotate(20deg) scale(0.8, 1.4) skew(5deg, -20deg);
        -o-transform: rotate(20deg) scale(0.8, 1.4)  skew(5deg, -20deg);
        transform: rotate(20deg) scale(0.8, 1.4)  skew(5deg, -20deg);
}
```

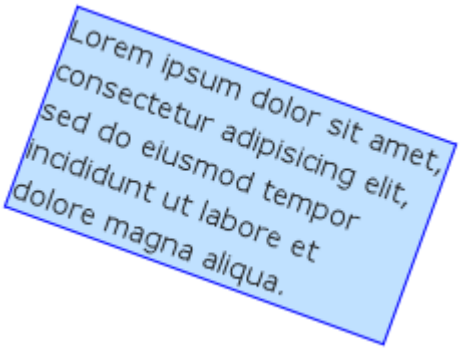

Figura 1 rotate(20deg)

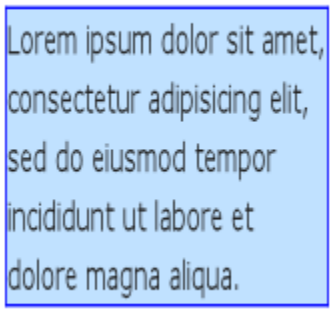

Figura 2 scale(0.8, 1.4)

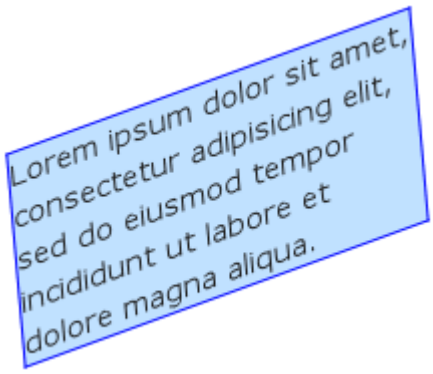

Figura 3 skew(5deg, -20deg)

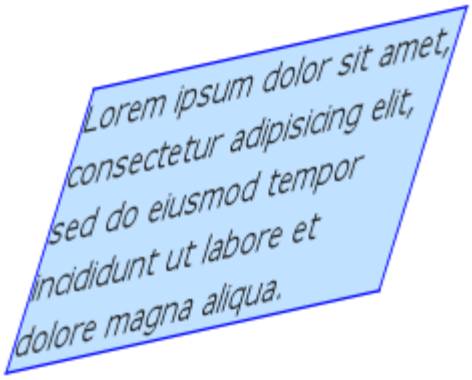

Figura 4 rotate(20deg) scale(0.8, 1.4) skew(5deg, -20deg)

Transform-origin

L'origine delle trasformazioni, se non specificato, corrisponde al centro dell'elemento. Ovvero i valori preimpostati della proprietà `trasform-origin` sono 50% e 50%. Il primo valore indica lo scostamento verso destra lungo l'asse delle x, il secondo lo scostamento verso il basso lungo l'asse y. Ad esempio la coppia di valori (0%, 0%) sposta l'origine della trasformazione nell'angolo in alto a sinistra, mentre la coppia di valori (100%, 100%) lo sposta nell'angolo in basso a destra. In alternativa si possono assegnare valori testuali: `left`, `center`, `right` per l'asse X e `top`, `center`, `bottom` per l'asse Y.

Esempio transform-origin css

```
#contenitore{
        width:200px;
        border: 1px solid blue;
        background: #c0e1ff;
        transform: rotate(40deg);
        transform-origin: left bottom;
        -ms-transform: rotate(40deg);
        -ms-transform-origin: left bottom;
        -moz-transform: rotate(40deg);
        -moz-transform-origin: left bottom;
        -webkit-transform: rotate(40deg);
        -webkit-transform-origin: left bottom;
        -o-transform: rotate(40deg);
        -o-transform-origin: left bottom;
}
```

Tabella di compatibilità

Bordi e sfondi					
`transform`	9.0+	3.5+	3.1+	4.0+	10.5+
`transform-origin`	9.0+	3.5+	3.1+	4.0+	10.5+

16 Transizioni e animazioni

Le transizioni e le animazioni rivoluzionano gli effetti grafici ottenibili tramite CSS. Tramite questo linguaggio dichiarativo le pagine web potranno spogliarsi di numerosi script, anche se per un certo periodo si dovrà ancora fare affidamento su JavaScript come modalità alternativa. Attualmente sono supportate da tutti i principali browser. Per le versioni più recenti di FireFox, Internet Explorer e Opera i prefissi proprietari non sono richiesti.

Transizioni

Le transizioni di CSS 3 sono definite all'interno del modulo *CSS3 Transitions*. Con questa innovazione è possibile cambiare lo stile di un elemento ottenendo un effetto di transizione non istantaneo, ovvero è stata introdotta la possibilità di cambiare gradualmente il valore assegnato da una o più proprietà, passando da uno stato inziale a un altro finale in un determinato intervallo di tempo.

Le componenti essenziali di un processo di transizione si potrebbero schematizzare nel seguente modo:

- La proprietà alla quale applicare la transizione.
- Per tale proprietà definire due valori: quello iniziale, contenuto nello stesso stile in cui viene definita la transizione, e quello finale, contenuto in un secondo stile.
- Un evento che determina l'inizio della transizione: nel momento in cui l'evento viene scatenato, se ad un elemento viene applicato un nuovo stile contenente la proprietà definita per la transizione, quest'ultima comincia il passaggio graduale dal valore iniziale a quello finale.
- La durata della transizione ovvero il tempo che intercorre tra il passaggio dal valore inziale a quello finale della proprietà prescelta.

Le proprietà da utilizzare per definire una transizione sono le seguenti:

transition-property

Il valore corrisponde alla proprietà alla quale applicare la transizione. Nel caso in cui si volesse applicare la transizione a più proprietà, queste devono essere assegnate a `transition-property` un'unica volta utilizzando la virgola come separatore.

transition-duration

Il valore di questa proprietà rappresenta l'intervallo temporale che intercorre tra il passaggio dal primo al secondo valore. Nel caso in cui le proprietà della transizione siano più di una, si può impostare l'intervallo per ognuna di esse utilizzando la virgola come separatore.

transition-delay

Questa proprietà non è obbligatoria per la definizione di una transizione. Il suo valore definisce il ritardo (espresso in secondi) dopo il quale la transizione avrà inizio.

transition-timing-function

Questa proprietà non è obbligatoria per la definizione di una transizione, definisce le modalità di transizione nell'intervallo prestabilito. Si possono applicare sei valori:

- `ease`: il processo di transizione rallenta gradualmente. Questo è il valore preimpostato.

- `linear`: il processo di transizione procede in modo costante.
- `ease-in`: la partenza viene accelerata.
- `ease-out`: l'arrivo viene rallentato.
- `ease-in-out`: partenza accelerata con arrivo rallentato.
- `cubic-bezier(x1, y1, x2, y2)`: è una funzione che rappresenta una curva di Bézier, permette di specificare qualsiasi tipo di andamento. I valori descritti prima sono tutti un'implementazione di questa curva.

transition

Questa proprietà permette di definire con un'unica assegnazione tutte le proprietà descritte. L'ordine di assegnazione è il seguente: `transition-property transition-duration transition-timing-function transition-delay`.

Esempio transizione

```
#contenitore{
        width:300px;
        height: 300px;
        background:green;
        -webkit-transition: -webkit-transform 2s ease-out 1s, background
        1s ease-in;
        -o-transition: -webkit-transform 2s ease-out 1s, background 1s
        ease-in;
        -moz-transition: -webkit-transform 2s ease-out 1s, background 1s
        ease-in;
        transition: -webkit-transform 2s ease-out 1s, background 1s
        ease-in;
}
#contenitore:hover{
        background: red;
        -webkit-transform: rotate(180deg) scale(-1.5, -1.5);
        -o-transform: rotate(180deg) scale(-1.5, -1.5);
        -moz-transform: rotate(180deg) scale(-1.5, -1.5);
        -ms-transform: rotate(180deg) scale(-1.5, -1.5);
        transform: rotate(180deg) scale(-1.5, -1.5);
}
```

Animazioni

Le animazioni sono rappresentate da un modulo slegato dalla definizione delle transizioni, però presentano molte affinità con quest'ultime. In realtà sono un'estensione delle transizioni, infatti, entrambe modificano nel tempo i valori delle proprietà CSS. La differenza è che le transizioni sono eseguite nel momento in cui i valori delle proprietà cambiano, mentre le animazioni sono eseguite nel momento in cui le proprietà di animazione sono applicate.

@keyframes

Questa regola permette di definire diversi passaggi di un'animazione. Ogni passaggio è rappresentato da una percentuale che corrisponde allo scostamento sull'asse temporale dell'intera animazione. All'interno dei vari passaggi si possono definire le proprietà ed i relativi valori, che saranno assegnati gradualmente fino a raggiungere il valore prefissato nell'istante definito. Tramite la proprietà `aniamtion-timing-function`, analoga alla proprietà associata alle transizioni `transition-timing-function`, si può inoltre specificare la modalità di esecuzione da un passaggio al successivo.

Esempio @keyframes css

```
@-webkit-keyframes animazione{
        0%{
                background:red;
                -webkit-aniamtion-timing-function:linear;
        }
        50%{
                background:green;
                -webkit-aniamtion-timing-function:ease-out;
        }
        100%{
                background:blue;
        }
}
```

La regola `keyframes` e la proprietà `aniamtion-timing-function` attualmente devono essere precedute dai prefissi proprietari per poter essere interpretate. Nell'esempio è riportato solamente il prefisso associato a Google Chrome e Safari. Il nome seguito dalla regola `keyframes` rappresenta l'identificativo univoco dell'animazione. In alternativa i valori `0%` e `100%` si possono sostituire rispettivamente con `from` e `to`.

Per applicare un'animazione a uno stile CSS ed impostarne i parametri si utilizzano le seguenti proprietà:

animation-name

Il valore dev'essere uguale all'identificativo dell'animazione che si vuole associare. Nel caso in cui si volessero applicare più animazioni, queste devono essere assegnate a `animation-name` un'unica volta utilizzando la virgola come separatore.

animation-duration:

Il valore di questa proprietà rappresenta l'intero intervallo temporale espresso in secondi dell'animazione. Nel caso in cui le animazioni associate siano più di una, si può impostare l'intervallo per ognuna di esse utilizzando la virgola come separatore.

animation-iteration-count

Rappresenta il numero di esecuzioni di ogni animazione. Se questa proprietà non è impostata, il suo valore è 1. Oltre ai numeri interi, si può impostare il valore `infinite` che crea un ciclo infinito.

animation-direction

Questa proprietà, se impostata a `reverse`, permette l'esecuzione dell'animazione in ordine inverso. Un altro valore disponibile è `alternate` che permette l'esecuzione delle iterazioni pari dall'inizio verso la fine, mentre quelle dispari in ordine inverso. Se non impostata, il suo valore è `normal` che corrisponde a un'esecuzione dall'inizio verso la fine.

animation-delay

Questa proprietà non è obbligatoria per la definizione di un'animazione. Il suo valore definisce il ritardo (espresso in secondi) dopo il quale il processo avrà inizio.

aniamtion-timing-function

Se questa proprietà non è impostata all'interno della regola `keyframes`, specifica la modalità di esecuzione da un passaggio al successivo. I valori disponibili sono analoghi a quelli di `transition-timing-function` riguardante le transizioni.

aniamtion-fill-mode

Questo attributo consente di definire lo stile dell'elemento interessato prima e dopo l'animazione. Può assumere il valore di `backwards` nel caso si debba applicare lo stato del primo `keyframe` anche prima dell'esecuzione, `forward` nel caso si debba applicare lo stile dell'ultimo `keyframe` anche dopo l'animazione. Nel caso si volesse scegliere entrambe le opzioni, il valore da assegnare è `both`.

Animation

Questa proprietà permette di definire con un'unica assegnazione tutte le proprietà descritte. L'ordine di assegnazione è il seguente: `animation-name animation-duration animation-timing-function animation-delay animation-iteration-count animation-direction aniamtion-fill-mode`.

Esempio animazione css

```
#contenitore{
        width: 300px;
        height: 300px;
        background:blue;
        -webkit-animation: animazione 3s infinite alternate;
        -o-animation: animazione 3s infinite alternate;
        -moz-animation: animazione 3s infinite alternate;
        animation: animazione 3s infinite alternate;
}
```

Compatibilità

Transizioni ed animazioni					
`transition`	10.0+	4.0+	3.1+	4.0+	10.5+
`animation`	10.0+	5.0+	4.0+	4.0+	12.0+

17 Un esempio pratico

In allegato alla tesi è presente un sito web con il fine di mettere in pratica alcune delle novità introdotte da HTML 5 e CSS 3. Di seguito descriverò solamente i contenuti associati a queste due nuove tecnologie.

Font

Lo stile del testo utilizzato è *ColabLig* scaricato dal sito web http://www.fontsquirrel.com/ ed è stato assegnato tramite la direttiva `@font-face` presente nella specifica CSS 3. Per motivi di compatibilità tra i vari browser sono stati inseriti diversi formati dello stile in questione. Tutti i documenti necessari all'inclusione di tale stile sono presenti all'interno della cartella *font*.

Testata

La testata della pagina è delimitata dal nuovo elemento semantico `<header>`. Lo sfondo è stato definito tramite la proprietà CSS 3 `radial-gradient`. Questa proprietà richiede i prefissi proprietari. Per i browser che non supportano il gradiente, è stata assegnata la proprietà `background-color` con il fine di evitare uno sfondo totalmente bianco.

All'interno della testata è presente un menù di navigazione delimitato dall'elemento `<nav>`. Ogni voce del menu corrisponde a una pagina del sito web. Nel momento in cui si visualizza una determinata pagina, viene associato un determinato stile alla relativa voce di menù. Il risultato di questo stile è dato dalle nuove proprietà `linear-gradient`, `border-radius` e `box-shadow`, prima ottenibile solamente applicando delle immagini di sfondo. Per i browser che non supportano il gradiente, sarà applicato uno sfondo bianco.

Un'altra novità della testata è rappresentata dal riflesso dell'immagine sulla destra tramite la proprietà box-refect. Per il momento è supportata solamente da Google Chrome.

L'ultima novità riguardante la testata è l'animazione del titolo e dell'immagine, visibile, per scelta, solamente nell'home page. La durata complessiva dell'animazione è di tre secondi e si può suddividere in due parti:

- La prima parte dura un secondo nel quale il titolo ed il sottotitolo assieme al paragrafo scorrono rispettivamente da destra e da sinistra fino a raggiungere la posizione finale. Queste animazioni sono identificate da `@keyframes dx` e da `@keyframes sx`.
- La seconda parte dura due secondi nei quali compare gradualmente l'immagine principale. Questa animazione, identificata da `@keyframes op`, ha un ritardo di un secondo in più rispetto alla prima, con lo scopo di evitare una sovrapposizione tra testo e immagine.

Piè di pagina

Nel piè di pagina è stato applicato un gradiente lineare nella sezione in cui è presente il menù di navigazione e un gradiente radiale come sfondo delle note finali (analogo a quello della testata ma con origine opposto).

Corpo centrale

Anche qua è stato applicato un leggero gradiente lineare. In questo caso però sono stati definiti tre colori di stop: il primo(0%) e l'ultimo(100%) bianco e il centrale(50%) grigio chiaro.

Home page

Nell'home page nella classe `commonBoxHome header` è stata utilizzata la proprietà `box-shadow`. In particolare il valore di tale proprietà contiene la parola chiave `inset` che applica l'ombra all'interno dell'elemento.

Le "sfere" dell'elenco puntato delle news, sono state create assegnando a un `<div>` un gradiente radiale a due colori con origine `center center` e impostando la proprietà `border-radius` a 50%.

Chi siamo

L'unica particolarità di questa pagina è rappresentata dalla posizione delle immagini, attribuita in modo dinamico tramite il selettore `nth-of-type()`. In sintesi, le immagini con indice dispari vengono posizionate sulla sinistra tramite la classe `#aboutUsImg section:nth-of-type(odd) img`, mentre quelle con indice pari sulla destra tramite la classe `#aboutUsImg section:nth-of-type(even) img`.

Ricerche

Per ogni ricerca è stato utilizzato un gradiente lineare con gli stessi colori di stop di quello associato al corpo centrale, assegnati con ordine opposto.

La data di ogni ricerca è rappresentata dal nuovo elemento HTML `<time>`.

Tramite l'utilizzo del selettore `:hover`, già presente nella specifica CSS 2.1, nel momento in cui si sposta il puntatore sopra una ricerca, viene applicata una transizione, della durata di 0,5 secondi, alle seguenti proprietà:

- `transform`: tramite la funzione `scale()` il contenuto viene scalato lungo entrambi gli assi di un fattore 1,2.
- `box-shadow`: compare un alone blu attorno al contenitore.

News

Nell'elenco puntato di sinistra sono definiti una serie di collegamenti ipertestuali associati alle news e per ognuno di essi l'attributo `href` è stato impostato con le cosiddetta "ancora" associata alla news corrispondente. Con questa impostazione, se la news non è selezionata, la classe associata è `newsDetails`. Nel momento in cui si clicca un collegamento ipertestuale, alla news in questione viene associata la classe CSS `newsDetails:target`. Nel primo caso, tramite l'utilizzo della proprietà `opacity` impostata a 0, la news è completamente

trasparente. Nel secondo caso, applicabile solamente ad una news in un determinato istante, quest'ultima sarà visibile. Le transizioni presenti in questa pagina si possono associare a due gruppi e vengono invocate tramite il nuovo selettore CSS 3 :target. Il primo gruppo è quello definito nelle classi in relazione con la classe `newsDetails` che definiscono la scomparsa di tutti gli elementi associati alla news. Il secondo è definito nelle classi in relazione con `newsDetails:target` nelle quali è definita la comparsa delle news. Nel momento in cui si clicca un collegamento, entrambi i gruppi di transizioni vengono eseguiti ma il secondo presenta un ritardo di 1 secondo per permettere il completamento del primo gruppo di transizioni.

Contatti

In questa pagina il form dei contatti è stato sviluppato utilizzando i nuovi *input type* di HTML 5 in modo tale da facilitare l'accessibilità all'utente. Tutti i campi contengono l'attributo `placeholder` che permette di visualizzare il suo valore all'interno dello spazio dedicato alla compilazione, prima di attivare il `focus` sull'elemento in questione. Inoltre alcuni campi contengono l'attributo `required` che obbliga l'utente a compilare il campo prima dell'invio.

Conclusioni

HTML 5 e CSS 3 rappresentano una nuova frontiera delle applicazioni web. I nuovi elementi semantici di HTML 5 permettono una strutturazione del contenuto più intuitiva, riducendo così tempi e costi di manutenzione. I Nuovi Web Form semplificano il lavoro degli autori di pagine web e rendono più facile e gradevole l'interazione degli utenti con i contenuti. Video Audio e Canvas permettono di trasmettere i contenuti con più efficacia, poiché l'interpretazione è più immediata e tutto questo senza dover ricorrere a strumenti aggiuntivi come, ad esempio, Flash. Microdata Web Storage e Geolocation incrementano gli strumenti di gestione e presentazione tramite l'utilizzo di semplici funzioni JavaScript. Le nuove pseudo-classi di CSS 3 permettono di separare in maniera più incisiva contenuti e stili di presentazione. Inoltre sono state introdotte numerose proprietà per creare effetti prima realizzabili tramite tecniche più complesse. Le Transizioni e le Animazioni realizzano effetti grafici dinamici con poche righe di codice altrimenti ottenibili con complessi script o con tecnologie proprietarie come Flash.

I cicli di rilascio delle versioni dei browser più popolari si stanno riducendo, rendendo più rapido l'estendersi del supporto delle nuove funzionalità. Opera, nonostante sia il browser meno utilizzato, supporta la maggior parte delle innovazioni introdotte.

Quando queste due tecnologie saranno ampiamente diffuse, il web acquisirà una maggiore dinamicità ed efficacia nella presentazione dei contenuti.

Riferimenti bibliografici

- Libro: HTML 5 Guida operativa. Di Mark Pilgrim.
- Libro: HTML 5 e CSS 3. Di Gabriele Gigliotti.
- Sito web: Web Hypertext Application Technology Working Group http://www.whatwg.org/
- Sito web: World Wide Web Consortium http://www.w3.org/
- Sito web: http://www.html.it/
- Sito web: http://www.w3schools.com