smartpixel

Introduzione a HTML5

Guida pratica alle principali novità

Guardare al futuro del web

HTML5 è il **nuovo linguaggio di markup** attualmente in fase di definizione presso il W3C (World Wide Web Consortium): non è un linguaggio completamente nuovo e diverso dai precedenti, ma l'evoluzione dell'attuale HTML 4.01 (il cui sviluppo si è fermato al 1999) pensato per coesistere con XHTML 2 oggi largamente diffuso

HTML è un linguaggio semplice che però da tempo ha dimostrato sempre di più **numerosi limiti** nel fornire agli utenti funzionalità più adatte a rispondere agli utilizzi più recenti di Internet, specialmente sotto la spinta dei nuovi dispositivi (smartphone, tablet, ecc.) e dei nuovi servizi (social network, file sharing, cloud computing, ecc.)

Mentre il W3C aveva abbandonato lo sviluppo di HTML in favore della specifica XHTML, nel 2004 un gruppo di sviluppatori, poco soddisfatti della direzione di sviluppo intrapresa dal W3C, formarono il **WHATWG** (Web Hypertext Application Technology Working Group) per lavorare sulla specifica HTML5 con l'obiettivo di espandere il linguaggio HTML attraverso una serie di nuove caratteristiche rivolte allo **sviluppo delle applicazioni web** e al miglioramento della compatibilità e dell'interoperabilità delle informazioni su diversi browser

Principali caratteristiche

Con HTML5 – "*the only version of HTML designed for a web of applications, not just documents*" (Jeffrey Zeldman) - il browser diventerà un vero e proprio sistema operativo che farà funzionare una vasta gamma di applicazioni → le sue caratteristiche sono:

- il lungo e complesso **doctype** è stato sostituito con un semplice `<!doctype html>`
- **nuovi elementi** di controllo del layout e regole più stringenti per la strutturazione semantica e la separazione tra struttura, presentazione e comportamento
- gli elementi che hanno dimostrato scarso o nessun utilizzo sono **deprecati** o **eliminati**
- sono estesi a tutti i tag una serie di attributi, specialmente quelli finalizzati all'**accessibilità**
- migliorati gli elementi di controllo per i **moduli** e i **contenuti multimediali**
- l'elemento **canvas** permette di usare Javascript per creare animazioni e grafica vettoriale
- il **drag & drop** attiva la funzionalità di trascinamento
- introduzione della **geolocalizzazione** dovuta alla forte espansione di sistemi operativi mobili
- **html5storage**, sistema alternativo ai normali cookie per la memorizzazione locale di dati scaricati via browser, ma più efficiente per il notevole risparmio di banda, permetterà di usare applicazioni web based anche senza collegamento Internet

Struttura del documento

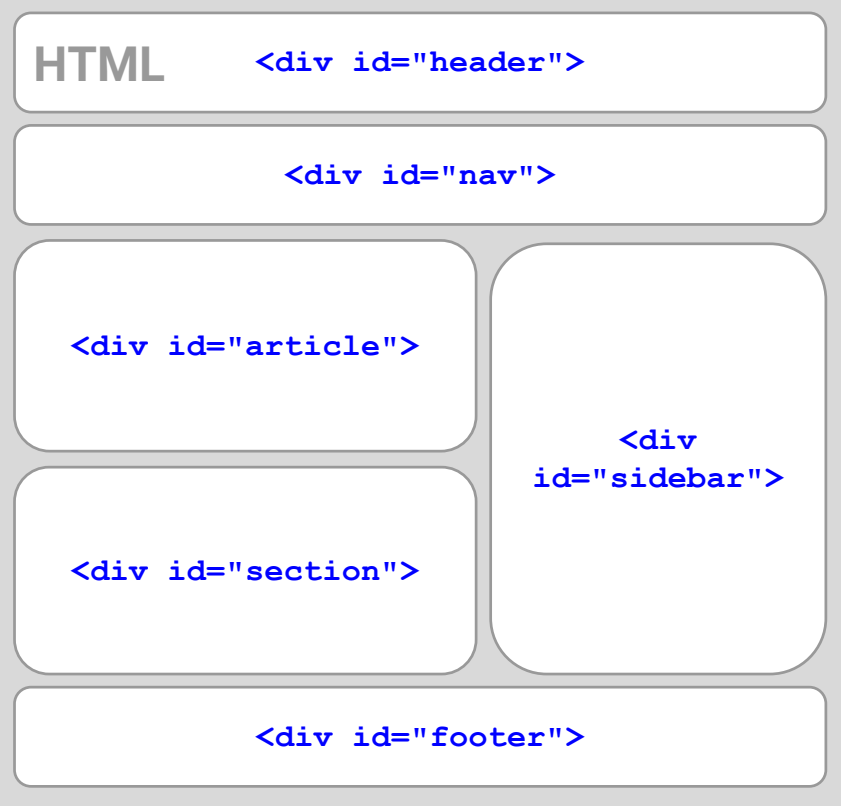
Dal punto di vista della **struttura del documento** le novità sono:

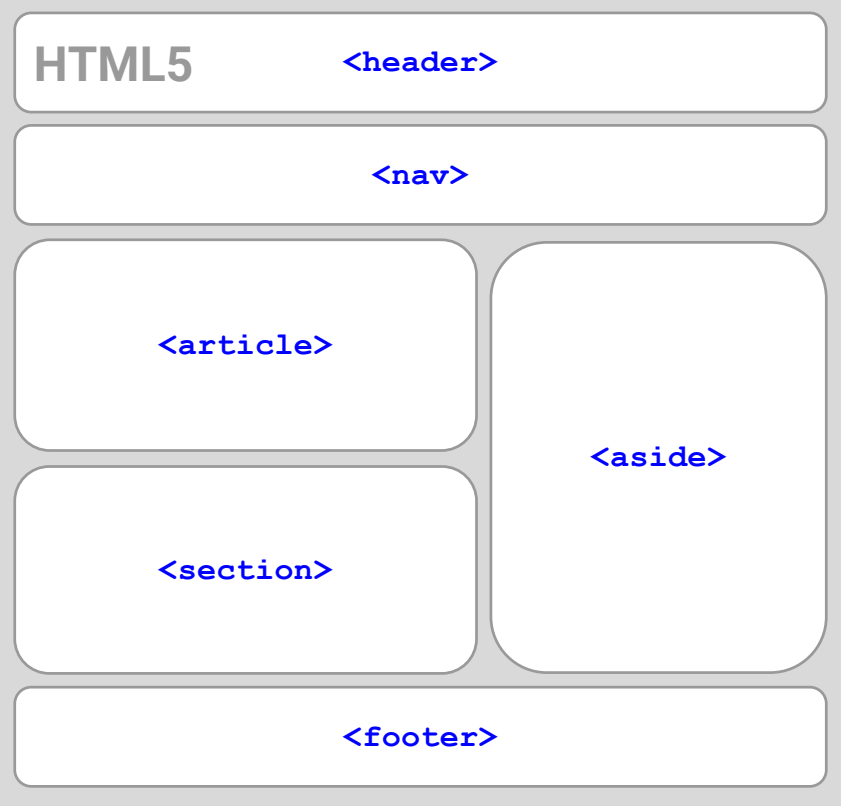
- lo **shorten doctype** o doctype dtdless `<!doctype html>` essendo privo di indicazioni sulla DTD (Document Type Definition) forza automaticamente il browser in modalità standard
- la **codifica dei caratteri** avviene specificando un charset valido per il documento (meglio in formato UTF-8, piuttosto che ISO) attraverso la sintassi `<meta charset="utf-8">` alternativa al classico `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">`
- HTML5 come tutti i linguaggi di markup presenta una **struttura di base** racchiusa dall'elemento radice <html> che comprende <head>, <title> e <body> per una corretta suddivisione delle informazioni e dei rispettivi contenuti
- per il **collegamento a file esterni** (script e CSS) si usano sempre i tag <script> e <style> ai quali però si può omettere l'attributo "*type*" (nei CSS, se si utilizza il tag <link> occorre aggiungere l'attributo "*rel*":

```
<script>// script interno</script>
<script src="js/javascriptesterno.js"></script>
<style>/* Foglio di stile interno */</style>
<link href="css/fogliodistileesterno.css" rel="stylesheet">
```

I nuovi elementi strutturali

HTML5 semplifica la scrittura del markup fornendo un **nuovo vocabolario semantico** che attribuisce significato a tutti quegli elementi che definiscono la struttura del documento e che erano resi attraverso i e gli attributi ID e di classe:





Il supporto da parte dei browser

Probabilmente HTML5 è l'ultima grande tecnologia per il web, ma **alcuni browser non hanno un supporto nativo** per i nuovi elementi semantici

Mentre la maggior parte dei browser supportano HTML5 (il browser individua l'elemento, lo aggiunge al DOM - Document Object Model - della pagina e gli attribuisce i valori di default o quelli definiti via CSS) **Internet Explorer < 8** non riconosce gli elementi HTML5 e li interpreta come nodi vuoti nel DOM, rendendo impossibile definire uno stile per l'elemento: perciò se è presente un elemento <aside> flottante a destra, IE non visualizzerà l' elemento così definito perché non è a conoscenza della sua esistenza

Esiste una **soluzione molto semplice** al problema, utilizzando Javascript è possibile aggiungere al DOM ogni elemento HTML5 per permetterne la modifica via CSS → perché funzioni occorre inserire lo script nella sezione <head> con un commento condizionale per Internet Explorer:

```
<!--[if IE]><script>document.createElement("header");</script><![endif]-->
```

Il supporto da parte dei browser

Considerato che i nuovi elementi dell'HTML5 sono numerosi forse è meglio adottare una **soluzione più comoda e sicura**: HTML5 enabling script di Remy Sharp, oltre a velocizzare la realizzazione dei documenti, considera anche problematiche più specifiche, come i CSS per la stampa, dove i Javascript non vengono eseguiti

```
<!--[if lte IE 8]><script
src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script><![endif]-->
```

Anche se ciò significa che attualmente le realizzazioni in HTML5 dipendono da Javascript, se è stato utilizzato markup semantico nonostante gli elementi non siano interpretati correttamente, i **contenuti restano completamente accessibili** (i contenuti degradano correttamente, similarmente a quanto accade se i CSS sono disabilitati)

Altri browser come **Firefox 3.6.x**, **Opera 10.60** e **Camino 1**, non riconoscendo i nuovi elementi HTML5, li posizionano accanto all'elemento precedente → la soluzione consiste nel normalizzare via CSS i nuovi elementi HTML5 definendoli come elementi block-level:

```
header, hgroup, nav, article, section, aside, footer  { display: block; }
```

Comprendere la semantica

HTML5 **rafforza la separazione** tra struttura e presentazione dei contenuti (l'aspetto di un titolo può variare graficamente, ma sotto il profilo strutturale resta sempre un titolo), occupandosi di definire la struttura dei contenuti a cui conferisce una maggiore standardizzazione semantica:

- descrive il **significato delle informazioni** attraverso una gamma di elementi semantici (tag e attributi) utilizzati per "*marcare*" le diverse strutture tipografiche (titoli, paragrafi, liste, tabelle, citazioni…) che identificano le informazioni, per renderle maggiormente accessibili
- definisce la **struttura dei contenuti**: i nuovi elementi HTML5 ordinano i contenuti a seconda dell'appartenenza semantica e NON del loro posizionamento nella pagina
- si presta ad un **uso modulare**: HTML5 modifica il normale modo di trattare la semantica degli elementi, che possono essere riutilizzati più volte all'interno della stessa pagina

Mentre finora l'unico modo per definire la struttura dei contenuti era di usare i titoli <h1-h6> ordinati gerarchicamente, in HTML5 i nuovi elementi strutturali definiscono sezioni indipendenti che permettono un utilizzo modulare degli elementi: <h1-h6> hanno valore di titolo di sezione e non più di pagina, in pratica `<header><h1>…` che marca il titolo del documento è ben diverso `<article><h1>…` che identifica il titolo dell'articolo

Comprendere la semantica

XHTML e **HTML** mancano dei necessari elementi semantici per descrivere con precisione il significato univoco degli elementi che formano la struttura del documento e richiedono di fare largo uso di opportunamente corredati di classi e ID → in **HTML5** i non sono più necessari per finalità strutturali, ma restano comunque validi nel caso in cui sia necessario introdurre dei contenitori per raggiungere specifici effetti di stile degli elementi, il cui significato viene però identificato dall'uso di precisi elementi di sezione, più adatti anche alle tecnologie assistive che aiutano l'utente a navigare più facilmente il documento:

- si può facilmente saltare la sezione di navigazione o passare direttamente da un articolo all'altro senza la necessità di fornire i classici skip link
- sostituendo molti dei con gli elementi semantici che identificano la struttura del documento si mantiene il codice più pulito, compatto e comprensibile

Per **dichiarare l'attributo** è sufficiente utilizzare la tradizionale formula nome=valore (`<html lang="it">`), in cui il valore dell'attributo può essere dichiarato a scelta senza apici, con apici singoli o doppi (è meglio usare sempre gli apici per mantenere più ordinato il codice)

Gli elementi strutturali: <header>

Secondo le specifiche l'elemento header rappresenta un'**introduzione** o aiuto per la navigazione: di solito racchiude i titoli (da <h1> ad <h6>, eventualmente racchiusi da <hgroup>), ma può contenere anche una tabella, un modulo di ricerca oppure il logo del sito web (a seguito di una recente modifica delle specifiche, è legittimo avere anche <nav> all' interno di header):

```
<header><h1>Titolo principale della pagina</h1></header>
```

L'elemento header (da non confondere con l'elemento <head>) non crea una nuova sezione, ma **è l'intestazione del sito**; inoltre può avere un utilizzo modulare che aggiunge un ottimo valore semantico per definire l'intestazione di una qualsiasi altra struttura, permettendo così di scrivere markup semantico, corretto e ordinato:

```
<article>
<header><h1>Titolo secondo articolo</h1></header>
<p>Lorem ipsum dolor set amet ...</p>
</article>
```

Gli elementi strutturali: <header>

All’interno di <header> l’elemento <hgroup> serve da **contenitore per un insieme di titoli** <h1-h6> eventualmente connessi:

- <hgroup> può contenere solo un gruppo di elementi <h1-h6>
- se l’intestazione consiste di un solo titolo marcato da <h1-h6>, non è richiesto <hgroup>
- se l’intestazione è composta da un titolo con sottotitolo/i, slogan o payoff (ad es. più di uno consecutivo a <h1-h6>), l’elemento <hgroup> raggruppa titolo e sottotitoli
- se l’intestazione è formata da un titolo con sottotitolo/i e metadata associati, <hgroup> raggruppa solo titolo e sottotitoli con <header> che contiene i metadata e <hgroup>

L’elemento <hgroup> **rafforza la struttura del documento**, può essere utilizzato in modo modulare per definire le intestazioni/titoli all’interno di diversi contesti senza che il significato venga confuso da elementi uguali presenti sullo stesso documento (<h1> in header>hgroup è diverso da <h1> in article>header>hgroup), usando <hgroup> la struttura dei titoli viene nascosta nell'outline dl documento tranne il titolo di primo livello del gruppo:

```
<header><hgroup><h1 … h6>Struttura di titoli e sottotitoli </h1 … h6></hgroup></header>
```

Gli elementi strutturali: <nav>

L'elemento <nav> è la sezione di pagina che **raggruppa i collegamenti ad altre pagine** (interne o esterne) ed è dedicata a raccogliere i link della navigazione principale fornendo un markup più rigoroso dal punto di vista semantico ed una struttura extra di grande aiuto per gli screenreader → oltre alla navigazione principale, l'elemento <nav> può essere usato anche per:

- **tabella dei contenuti**, in quanto navigazione principale per quel particolare tipo di contenuto
- **paginazione** pulsanti "*precedente/successivo*": anche in questo caso potrebbe essere importante marcarli con <nav> per la struttura complessiva e la gerarchia del sito
- **moduli di ricerca**: è estremamente importante per la navigazione di un sito, in particolare siti di grandi dimensioni che si basano quasi esclusivamente sul loro motore di ricerca interno
- **breadcrumbs**: sebbene non siano sempre necessari, nei siti di grandi dimensioni possono essere un importante aiuto alla navigazione

In HTML5 c'è anche il tag <menu> che però è un elemento interattivo utilizzato esclusivamente nelle applicazioni web per marcare una lista di comandi e in quanto tale non va confuso con l'elemento <nav> nel marcare la navigazione principale:

```
<nav><ul><li …>Elenco dei link della navigazione principale</ …li></ul></nav>
```

Gli elementi strutturali: <article>

L'elemento <article> rappresenta una **sezione di contenuto semanticamente indipendente** dal resto del documento (come ad es. un messaggio di un blog o un articolo di una rivista) ed è perciò adatto a raccogliere al suo interno i contenuti principali del sito/applicazione

L'**utilizzo modulare** dei tag HTML5 permette di organizzare l'elemento <article> in modo rigoroso dal punto di vista della semantica dei contenuti prevedendo al suo interno sia le intestazioni che i piè di pagina relativi al contenuto di quel determinato articolo:

```
<article>
<header>
<hgroup>
<h1>Titolo dell'articolo</h1>
<h2>Sottotitolo dell'articolo</h2>
</hgroup>
</header>
<p>Testo dell'articolo</p>
<footer>Piè di pagina dell'articolo</footer>
</article>
```

Gli elementi strutturali: <section>

L’elemento <section> rappresenta una **qualsiasi sezione** del documento/applicazione, di solito corredata da un titolo <h1-h6> o da un’intestazione (viene utilizzato al posto dei tradizionali), come ad esempio un capitolo di un libro:

```
<section id="presentazione">
<header>
<h1>Titolo della presentazione</h1>
</header>
<article>
<header>
<hgroup>
<h1>Titolo dell'articolo</h1>
<h2>Sottotitolo dell'articolo</h2>
</hgroup>
</header>
<p>Testo dell’articolo</p>
<footer>Piè di pagina dell’articolo</footer>
</article>
</section>
```

Gli elementi strutturali: <aside>

L’elemento <aside> definisce una **porzione di codice correlata al contenuto principale**, dal quale è separata pur evidenziadone specifici punti di interesse: anche se inizialmente vincolato al contesto del solo elemento <article>, in seguito le specifiche hanno accolto il feedback degli sviluppatori permettendo che <aside> venga usato come contenuto secondario generico:

- se è **nidificato all'interno di <article>**, il contenuto deve essere specifico a tale articolo (ad esempio, note, suggerimenti, citazioni, glossari o elenchi di link correlati)
- se è **utilizzato come contenitore di qualcosa** che si trova al di fuori del flusso dell’articolo, il contenuto deve essere relativo all’intero sito (ad es. moduli di ricerca, link di navigazione aggiuntiva ed ogni contenuto pubblicitario se il contesto è relativo alla pagina)

Ogni contenuto che non è l'oggetto principale di un articolo/pagina ma è legato, correlato all'articolo/pagina, può essere correttamente marcato con <aside> che ha un forte significato semantico come **stand-alone**, cioè elemento non essenziale al contenuto, ma che può dare un ulteriore livello di informazione ai contenuti:

```
<aside>contenuto relazionato ma indipendente</aside>
```

Gli elementi strutturali: <footer>

L’elemento <footer> rappresenta la parte inferiore per i suoi elementi più vicini, cioè la sezione dei contenuti o dell'elemento radice → di solito contiene le **informazioni sul sito** e i riferimenti correlati, come il nome dell’autore (contatti), commenti, tag e categorie, copyright e simili:

```
<footer>Make with love and HTML5!</footer>
```

Un recente aggiornamento delle specifiche permette di dare all'elemento footer lo stesso modello di contenuto degli elementi <header> e <nav> con l’avvertenza che il footer rappresenta il **piè di pagina della sezione a cui si applica** → in altre parole, ha un utilizzo modulare che permette di utilizzare footer multipli, ciascuno dei quali diventerà l’elemento <footer> per quella specifica sezione del documento, che avrà lo scopo semantico di poter essere utilizzato più volte per specificare informazioni supplementari

L’elemento <footer> offre la possibilità di definire i documenti attraverso un markup rigoroso dal punto di vista semantico: è pertanto essenziale usare questi nuovi elementi strutturali per lo scopo per il quale sono nati ed evitare un utilizzo scorretto degli elementi di markup, come è già accaduto nel passato nel caso delle tabelle e dei

Elementi semantici di blocco

Una nuova feature HTML5 è quella di inscrivere **elementi di blocco** all'interno di link, attraverso la tecnica dei compound HTML, cioè strutture composte in grado di fornire significato più preciso di quanto sia possibile fare applicando un singolo elemento

Ad es. se un unico collegamento **<a>** contiene abstract troncato di un articolo, immagine e link per leggere il resto dell'articolo, si può creare un'area linkata molto più ampia in cui il testo alternativo dell'immagine fornisce maggiori informazioni agli screenreaders:

```
<article><a href="/story1.html" title="descrizione articolo">
<h3>Titolo dell'articolo</h3>
<p><img src="/img.jpg" alt="descrizione immagine" />Paragrafo di testo</p>
<p>Read more</p>
</a></article>
```

Altro elemento block-level è **<figure>**, contenitore di immagini/audio/video con didascalia:

```
<figure>
<content><img src="/images/audir8.jpg" alt="Audi R8" /></content>
<legend>Figura 1. Una splendida Audi R8</legend>
</figure>
```

La gestione dei moduli

I **<form>** hanno subito una importante rivoluzione → **nuovi<input type =“”>**:

- **datetime** e datetime local per l’inserimento della data e dell’ora anche in formato locale
- **number** per l’inserimento di cifre e numeri
- **range** verifica il valore del campo <input> all’interno di un intervallo
- **search** per i campi <input> dedicati a svolgere ricerche
- **email** conferma che il valore del campo <input> deve essere un indirizzo email valido
- **url** assicura che il valore del campo <input> sia un indirizzo web
- **color** fornisce un meccanismo che permette all’utente di specificare il colore in RGB

Tra gli **attributi**, i più interessanti sono:

- **required** rende il campo obbligatorio e costringe a inserire un valore per validare i dati
- **autofocus** permette di scegliere quale elemento è attivo al caricamento della pagina
- **placeholder** attiva la funzionalità di segnaposto nei campi modulo
- **pattern** costringe ad inserire un valore in un formato specifico come ad es. un codice postale

HTML5 & microformati

I **microformati** sono una parte di markup che contiene espressioni semantiche per renderle riconoscibili da parte dei browser, pur mantenendo un'elevata comprensibilità da parte delle persone: l'idea è che i browser possano esaminare ed utilizzare le informazioni marcate dai microformati (si parla a tal proposito di web semantico)

L'elemento **<address>** ha lo scopo di fornire informazioni di contatto ed era già presente nelle specifiche HTML 3 del 1995 → le specifiche HTML5 ne rafforzano il significato prevedendo che l'elemento <address> non deve essere utilizzato per contenere qualsiasi tipo di indirizzo, a meno che non siano informazioni di contatto rilevanti in un documento o in parte di esso:

```
<footer>
<div class="vcard"> by
<address class="author">
<em class="fn"><a href="www.jackosborne.com" title="Posts by Jack Osborne" href="#">Jack
Osborne</a></em>
</address> on
<time datetime="2009-11-04" class="published updated">November 4th, 2009</time>
</div>
</footer>
```

HTML5 & microformati

L’elemento <address> può comprendere i nomi degli autori, collegamenti a siti web correlati, indirizzi e-mail per commenti, indirizzi postali, numeri di telefono ecc., ma non è appropriato per gli indirizzi che non sono strettamente correlati al documento, i quali possono essere marcati attraverso l’elemento <p> in combinazione con il **microformat hCard**:

```
<div class="vcard">
<p class="fn"><a class="url" href="www.jackosborne.com">Dr. Jack Osborne</a><p>
<p class="adr">
<span class="street-address">HTML5 Hospital</span>
<span class="region">Doctorville</span>
<span class="postal-code">Postal Code</span>
<span class="country-name">Great Britain</span>
</p>
<p class="tel">+44 (0)XXXX XXXXXX</p>
</div>
```

Le Application Programming Interface (API)

Una novità molto importante di HTML5 è costituita dall'**introduzione delle API** (Application Programming Interface) per facilitare la creazione di applicazioni web:

- l'API **audio** e **video** facilita l'inserimento di contenuti multimediali
- l'API **drag & drop** attiva un sistema di trascinamento per cui certi oggetti (di default immagini e link) sono trascinati e poi rilasciati (richiede Javascript per funzionare pienamente)
- il metodo **getElementsByClassName** funziona come l'analoga getElementsById, andando a rintracciare e individuare tutti gli elementi a cui sia stata assegnata una specifica classe
- il modulo **html5storage** rende possibile la memorizzazione locale dei dati e permette di utilizzare le applicazioni web offline (anche dopo la chiusura della connessione Internet)
- la **geolocalizzazione** è una API molto potente: tramite il browser viene rilevata la posizione dalla quale ci si collega per eventualmente fornire dati personalizzati (per ragioni legate alla privacy i browser richiedono in automatico l'autorizzazione di questa funzione)
- **canvas** definisce un'area di disegno vettoriale programmabile con HTML5 e Javascript, dove poter visualizzare forme geometriche, immagini e testo, usare gradienti e ombre, effettuare operazioni di traslazione e rotazione, ecc.

Gli elementi multimediali: <video>

HTML5 offre la possibilità di inserire video all'interno delle pagine web permettendone la visione senza Flash Player → un video funziona grazie ad un contenitore (mpeg4, flash video, ogg, ecc.), i codec audio (mpeg1, vorbis, ecc.) e video (mpeg4 asp, h.264, theora), ma poiché **nessun codec è definito nelle specifiche**, attualmente è in corso una battaglia che contrappone i maggiori produttori di browser che utilizzano diversi codec per il rendering dei video HTML5

Opera e Mozilla usano il codec open source **Ogg Theora**, mentre Apple e Google usano il codec proprietario **H.264**, come farà molto probabilmente anche Microsoft, licenziataria assieme ad Apple del codec H.264: la speranza è che i siti leader del mercato come YouTube e Vimeo possano implementare codec crossbrowser interoperabili e - a giudicare dai relativi blog - sembra che entrambi stiano sviluppando diversi progetti in questa direzione

In attesa di vedere da quale parte penderà l'ago della bilancia, la scelta migliore è di **preparare due files**, uno con theora, vorbis e ogg (supportato da Firefox, Opera e Chrome) e il secondo con H.264, AAC e mp4 (supportato da Chrome, Safari, iPhone e Android): per la conversione si possono usare ffmpeg2theora e handbrake

Gli elementi multimediali: <video>

Il tag **<video>** permette di integrare il filmato specificando il comportamento del contenuto attraverso seguenti attributi:

```
<video src="clip.ogv" width="640" height="480" controls autoplay poster="poster.jpg">
<a href="clip.ogv">Download Movie</a>
</video>
```

- occorre specificare altezza e larghezza perché <video> si comporta come il tag <img>
- l'attributo "*controls*" rende visibili i controlli del video
- l'attributo "*autoplay*" fa partire automaticamente il filmato
- l'attributo opzionale "*poster*" può essere usato per specificare un'immagine che sarà visualizzata al posto del video prima che questo inizi
- se il video deve essere scaricato non appena la pagina è caricata è sufficiente aggiungere l'attributo "*preload*" o in caso contrario basta specificare "none" come valore

Gli elementi multimediali: <video>

L'attributo "*src*" dell'elemento <source> individua la risorsa specificando file e codec così il browser riconosce il formato di file e lo indirizza a specifici dispositivi tramite l'attributo "*media*", infine va previsto un contenuto alternativo per i browser che non supportano <video>:

```
<video controls autoplay poster="poster.jpg">
<source src="clip.3gp" type='video/3gpp; media="handheld"'>
<source src="clip.m4v" type='video/mp4; codecs="avc1.42E01E, mp4a.40.2"'>
<source src="clip.ogv" type='video/ogg; codecs="theora, vorbis"'>
<p>Spiacente il tuo browser non supporta l'elemento video</p>
</video>
```

Per controllare l'interfaccia in modo che possa adeguarsi al design del sito, l'API fornisce diversi metodi ed eventi per controllare il playback → i più semplici da usare sono "*play()*" e "*pause*", mentre "*currentTime*" riavvolge tutto dall'inizio:

```
<video src="clip.ogg" id="video"></video>
<script> var video = document.getElementById("video");</script>
<button type="button" onclick="video.play();">Play</button>
<button type="button" onclick="video.pause();">Pause</button>
<button type="button" onclick="video.currentTime = 0;">Rewind</button>
```

Per saperne di più

http://www.whatwg.org/specs/web-apps/current-work/multipage/ (Specifiche HTML5)

http://www.whatwg.org/ (WHATWG)

http://validator.w3.org/ (Servizio ufficiale di validazione del markup del W3C)

http://validator.nu/ (Servizio sperimentale di validazione HTML5)

http://microformats.org (Microformati)

Per imparare HTML5 attraverso esempi, demo, manuali online:
http://html5doctor.com/
http://html5demos.com/
http://diveintohtml5.org/