Le novità di HTML5

Il panorama di Internet è cambiato molto dall'assunzione a 'W3C Reccomandation' della versione precedente delle specifiche, avvenuta verso la fine del 1999. In quel tempo **il Web era strettamente legato al concetto di ipertesto** e l'azione più comune per l'utente era la fruizione di contenuti, tipicamente in forma testuale. La mediamente bassa velocità di connessione e il limitato investimento sul media contribuivano ad una scarsa presenza di applicazioni web, più care da sviluppare ed esigenti in termini di banda.

Tutto questo era ben rappresentato da un linguaggio, HTML, principalmente **orientato ad agevolare la stesura di semplici documenti testuali collegati fra loro**. Negli anni successivi l'interesse intorno alla rete ha subito una brusca accelerazione e questo ha condizionato positivamente sia la diffusione che la velocità di connessione della stessa, attirando di conseguenza maggiori investimenti e ricerca. Al modello di fruizione dei contenuti si è aggiunta la possibilità per l'utente finale di divenire esso stesso creatore attraverso **applicazioni web sempre più elaborate ed interessanti**.

Questo nuovo livello di complessità, in termini di sviluppo, ha però dovuto scontrarsi con un set di specifiche poco inclini ad essere utilizzate per tali fini e che quindi si sono prestate al compito solo a scapito di infiniti hack e workaround. Esempi di questi 'utilizzi non premeditati' dell'HTML si possono trovare un po' ovunque, famoso il caso degli attributi `rel` e `ref` che hanno assunto nel tempo valori non previsti, (eg: `nofollow`) anche esterni alla loro funzione naturale (eg: l'utilizzo di questi due attributi in librerie come Lightbox).

Parallelamente il percorso di crescita del web ha fatto emergere alcune **strutture di contenuto ricorrenti**, ben caratterizzate dal fenomeno dei blog: informazioni di testata, menu di navigazione, elenchi di articoli, testo a pie' di pagina, ed altri. La parte dedicata al singolo articolo presenta anch'essa solitamente lo stesso set di informazioni quali autore, data di pubblicazione, titolo e corpo del messaggio. Anche in questo caso l'HTML4 non ha saputo fornire gli strumenti adatti a consentire una corretta gestione e classificazione del contenuto obbligando gli sviluppatori web a ripiegare su strutture anonime, quali `<div>` e `<p>`, arricchite di valore semantico con l'utilizzo di attributi quali `class` e `id`.

L'HTML5 nasce per risolvere questi problemi offrendo agli sviluppatori web **un linguaggio pronto ad essere plasmato secondo le più recenti necessità**, sia dal lato della **strutturazione del contenuto** che da quello dello **sviluppo di vere e proprie applicazioni**.

Grandi cambiamenti nell'ombra

La differenza principale tra le versioni correnti di HTML e XHTML risiede nella sintassi. Il linguaggio di markup creato da Tim Berners-Lee, e che ancora oggi popola i nostri browser, è stato studiato come applicazione del più generico **SGML**, Standard Generalized Markup Language; ne è la prova la dichiarazione di Document Definition Type che dovrebbe essere posta nella prima riga di una pagina Web ad indicare la grammatica, HTML per l'appunto, usata nel documento:

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

In realtà la quasi totalità dei browser ignora la definizione e interpreta il documento secondo logiche più permissive, frutto di anni di eccezioni e di esperienza accumulata su pagine malformate. L'XHTML, invece, è una versione della sintassi HTML costretta all'interno delle regole XML, a sua volta grammatica SGML: questo approccio dovrebbe implicare un maggior rigore nella pagina e l'aderenza a regole quali l'obbligo di chiusura di tutti i tag. Il parser XML inoltre dovrebbe sospendere l'interpretazione della pagina al primo errore rilevato.

L'arrivo dell'HTML5 introduce una importante novità in questo scenario, per la prima volta l'obiettivo delle specifiche è quello di **definire un linguaggio ubiquo, che possa poi essere implementato su entrambe le sintassi**. L'occasione è buona anche per fare un po' di pulizia e rompere definitivamente il legame tra HTML e SGML formalizzando e traducendo in standard le regole adottate da tempo nei browser. Per indicare un documento HTML5 è nata quindi la seguente semplice istruzione:

```
<!DOCTYPE html>
```

Che si affianca a quella da utilizzare in caso si intenda scrivere una pagina XHTML5:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
```

Un nuovo content model

Ecco un esempio scritto utilizzando i nuovi elementi messi a disposizione da HTML5:

```
<!doctype html>
<html lang="it">
<head>
</head>
<body>
   <header>
   --- Titolo e Testata ---
   </header>
   <nav>
   --- Voci di Menu ---
   </nav>
   <article>
   --- Un Post ---
   </article>
   <article>
   --- Un altro Post ---
   </article>
</body>
</html>
```

Come si può notare, i tag introdotti hanno **un nome in strettissima attinenza con il proprio contenuto**; questo approccio risolve in modo elegante sia il problema dell'utilizzo dell'attributo `class` con valore semantico, sia la riconoscibilità delle singole aree del documento da parte di un browser. Ma non è tutto; l'introduzione di `article`, `nav`, `header` e altri tag che vedremo, impone anche sostanziose novità al modo in cui lo user-agent deve comportarsi nell'interpretare questi elementi.

Contenuti in una bolla di sapone

Partiamo dal seguente esempio HTML4:

```
<html>
<body>
   <h1>I diari di viaggio:</h1>
   <h2>A spasso per il mondo alla scoperta di nuove culture:</h2>
   <h3>Giro turistico della Bretagna</h3>
   <p>lorem ipsum..</p>
   <h3>Alla scoperta del Kenya</h3>
   <p>lorem ipsum..</p>
   <h3>Cracovia e la Polonia misteriosa</h3>
   <p>lorem ipsum..</p>
   <p>tutti i viaggi sono completi di informazioni su alberghi e
      prezzi</p>
</body>
</html>
```

Se lo visualizziamo avremo un risultato assolutamente strutturato come questo:

Figura 3 – Struttura del documento

Supponiamo ora di voler dividere i viaggi per continente. Con il modello attuale saremmo obbligati a cambiare l'elemento h3 in h4 in modo da fare spazio alla nuova suddivisione:

```
<html>
<body>
   <h1>I diari di viaggio:</h1>
   <h2>A spasso per il mondo alla scoperta di nuove culture:</h2>
   <h3>Europa</h3>
   <h4>Giro turistico della Bretagna</h4>
   <p>lorem ipsum..</p>
   <h4>Cracovia e la Polonia misteriosa</h4>
   <p>lorem ipsum..</p>
   <h3>Africa</h3>
   <h4>Alla scoperta del Kenya</h4>
   <p>lorem ipsum..</p>
   <p>tutti i viaggi sono completi di informazioni su alberghi e
      prezzi</p>
</body>
</html>
```

Questo accade perché la gerarchia delle intestazioni è assoluta rispetto all'intero documento e ogni tag <h*> è tenuto a rispettarla. Nella maggior parte dei casi però questo comportamento è fastidioso in quanto è molto comune avere a che fare con contenuti che, come articoli o commenti, vorremmo avessero una struttura indipendente dalla loro posizione nella pagina. In HTML5 questo è stato reso possibile definendo una nuova tipologia di content model, chiamato '**sectioning content**', al quale appartengono elementi come `article` e `section`. All'interno di tag come quelli appena citati la vita scorre come in una bolla di sapone, quindi l'utilizzo di un <h1> è considerato relativo alla sezione in cui si trova.

Riprendiamo l'esempio precedente ed interpretiamolo in salsa HTML5:

```
<!doctype html>
<html>
<head>
   <title>I diari di viaggio</title>
</head>
<body>
   <header>
      <hgroup>
         <h1>I diari di viaggio:</h1>
         <h2>A spasso per il mondo alla scoperta di nuove culture:</h2>
      </hgroup>
   </header>
```

```
   <section>
      <h1>Europa</h1>
      <article>
         <h1>Giro turistico della Bretagna</h1>
         <p>lorem ipsum..</p>
      </article>
      <article>
         <h1>Cracovia e la Polonia misteriosa</h1>
         <p>lorem ipsum..</p>
      </article>
   </section>
   <section>
      <h1>Africa</h1>
      <article>
         <h1>Alla scoperta del Kenya</h1>
         <p>lorem ipsum..</p>
      </article>
   </section>
   <p>tutti i viaggi sono completi di informazioni su alberghi e
      prezzi</p>
</body>
</html>
```

Molto meglio! Ora i singoli componenti di questo documento sono *atomici* e possono essere spostati all'interno della pagina senza dover cambiare la loro struttura interna. Inoltre, grazie a queste divisioni, il browser riesce a discernere perfettamente il fatto che l'ultimo paragrafo non appartenga al testo del viaggio in Kenia.

Diamo prova dell'atomicità creando un blocco dedicato all'ultimo articolo inserito: 'Un week-end a Barcellona':

```
<!doctype html>
<html>
<head>
   <title>I diari di viaggio</title>
</head>
<body>
   <header>
      <hgroup>
         <h1>I diari di viaggio:</h1>
         <h2>A spasso per il mondo alla scoperta di nuove culture:</h2>
      </hgroup>
   </header>
   <article>
      <h1>Un week-end a Barcellona</h1>
      <p>lorem ipsum..</p>
   </article>

<!-- resto della pagina -->
```

Anche grazie a questo content model l'HTML5 introduce un nuovo e preciso algoritmo per il calcolo dell'outline del documento. La vista ad outline, tipica nei software di word processing e ancora non presente nei browser, è utilissima nella navigazione dei contenuti di una pagina. Sperimentiamo questa feature installando un'apposita estensione per Chromium:

Figura 4 (click per ingrandire) – Vista ad outline del documento

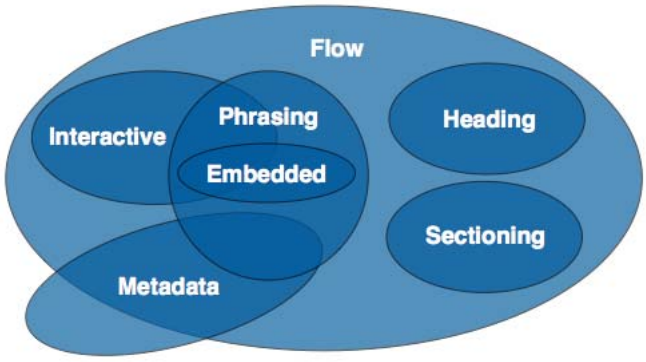
È interessante notare come l'algoritmo non solo identifichi correttamente i vari livelli di profondità, ma per ognuno di essi sappia anche recuperare il titolo adeguato. Nell'HTML5 è vitale che il design della pagina si rispecchi in una outline ordinata e coerente, questa infatti è la miglior cartina tornasole possibile in merito al corretto utilizzo delle specifiche: ad esempio, in una prima revisione della lezione, il codice HTML precedente mancava dell'elemento `hgroup`, utile a raggruppare elementi che concorrono a formare un titolo. L'errore è stato individuato e corretto proprio grazie alla visualizzazione di una outline errata:

Figura 5 – Strutturazione corretta del documento

Concludiamo la trattazione di questo content model citando la presenza di alcuni elementi che, pur seguendo le linee interpretative del 'sectioning content', devono essere ignorati dall'algoritmo di outline. Tali tag sono definiti '**sectioning roots**' ed il motivo della loro esclusione è legato al fatto che essi non concorrono tanto alla struttura dei contenuti della pagina quanto all'arricchimento della stessa. Fanno parte di questo elenco elementi come: `blockquote`, `details`, `fieldset`, `figure` e `td`. Seppur esclusi dall'outline del documento, nulla vieta agli appartenenti di questo gruppo di avere una propria outline interna.

Panoramica sui content model

HTML5 offre differenti tipologie di disposizione del contenuto:



Metadata content

Fanno parte di questa categoria tutti gli elementi utili alla **definizione del documento nel suo insieme**: a livello di presentazione o di funzionamento.

Tag: `base`, `command`, `link`, `meta`, `noscript`, `script`, `style`, `title`.

Sectioning content

Ne abbiamo appena parlato: il gruppo contiene **tutti quegli elementi studiati per ospitare contenuti** *atomici* **e semanticamente ordinati**. È importante utilizzare almeno un appartenente alla categoria heading content all'interno di ognuno di questi tag, questo anche per non avere un outline popolato da voci senza titolo (vedi immagine).

Tag: `article`, `aside`, `nav`, `section`.

Figura 6 – Visualizzazione della struttura del documento

Heading content

Tutti gli appartenenti a questo gruppo servono per **definire titoli**. Interessante notare come se la presenza di uno di questi elementi non sia associata ad un sectioning content questo venga definito implicitamente.

Tag: h1, h2, h3, h4, h5, h6, hgroup

Phrasing content

Incorpora il **testo del documento** così come tutti i **modificatori tipografici e visuali** dello stesso.

Tag: a (solo se contiene phrasing content a sua volta), `abbr`, `area` (se figlio di un elemento `map`), `audio`, `b`, `bdi`, `bdo`, `br`, `button`, `canvas`, `cite`, `code`, `command`, `datalist`, `del` (solo se contiene phrasing content a sua volta), `dfn`, `em`, `embed`, `i`, `iframe`, `img`, `input`, `ins` (solo se contiene phrasing content a sua volta), `kbd`, `keygen`, `label`, `map` (solo se contiene phrasing content a sua volta), `mark`, `math`, `meter`, `noscript`, `object`, `output`, `progress`, `q`, `ruby`, `s`, `samp`, `script`, `select`, `small`, `span`, `strong`, `sub`, `sup`, `svg`, `textarea`, `time`, `var`, `video`, `wbr`.

Embedded content

Ne fanno parte tutti quegli elementi che, come il nome del gruppo suggerisce, **incorporano nella pagina oggetti esterni**.

Tag: `audio`, `canvas`, `embed`, `iframe`, `img`, `math`, `object`, `svg`, `video`.

Interactive content

Questa categoria comprende tutto ciò specificatamente studiato per interagire **con l'utente**.

Tag: `a`, `audio` (con l'attributo controls presente), `button`, `details`, `embed`, `iframe`, `img` (con l'attributo usemap presente), `input` (con l'attributo `type` diverso da `hidden`), `keygen`, `label`, `menu` (con l'attributo `type="toolbar"`), `object` (con l'attributo `usemap` presente), `select`, `textarea`, `video` (i con l'attributo `controls` presente).

Come avrete notato ogni elemento può appartenere ad uno o più content models, anche a seconda della sua configurazione degli attributi. Oltre ai gruppi citati, che sono i principali, va ricordato **flow**, che raggruppa la quasi totalità degli elementi e alcuni gruppi creati specificatamente per i controlli dei form, che vedremo più avanti.

I nuovi elementi di formattazione della pagina HTML

Man mano che le esigenze di pubblicazione dei contenuti HTML sono mutate nel tempo, la formattazione delle pagine è diventata un aspetto via via sempre più complicato.

Alla formattazione semplice e lineare dei primi siti, è seguita successivamente la necessità di posizionare i contenuti in modo vario e complesso.

Inizialmente, le tabelle hanno rappresentato la soluzione al problema, fornendo un meccanismo per creare una struttura statica dove posizionare i contenuti all'interno delle pagine.

In seguito, data la molteplicità di browser e di dispositivi da supportare, la staticità propria delle tabelle si è rivelata un limite troppo forte. Pertanto, nei siti più moderni, si è cominciato ad usare gli elementi flottanti, ottenibili tramite l'impiego del tag `div` unitamente agli stili `float` e `clear`, dal momento che essi garantiscono un livello di flessibilità molto maggiore rispetto alle tabelle.

Il tag div rappresenta peraltro un elemento di valenza generale, senza una particolare connotazione. Esso può essere impiegato secondo modalità e finalità diverse, raggruppando altri elementi o contenuti in ogni punto all'interno del corpo della pagina.

HTML5 affronta il problema della formattazione delle pagine in modo radicale e presenta una serie di nuovi elementi di markup, destinati a fornire una soluzione alle esigenze di pubblicazione più comuni e a dare una connotazione semantica alle diverse parti della pagina.

Diversamente dal tag div, ciascuno di questi nuovi elementi nasce con un preciso scopo.

La tabella sotto riassume brevemente le finalità di ciascun tag.

Tag	*Descrizione*
`<article>`	**Rappresenta una sezione indipendente, contenente principalmente un contenuto testuale, ma non solo**
`<aside>`	**Rappresenta una sezione che include un contenuto che è collegato a quanto trattato nella pagina, ma che è comunque distinto da esso**
`<header>`	**Rappresenta un blocco di intestazione all'interno della pagina o di una sezione (elementi `section`, `article`, `aside` e nav)**
`<hgroup>`	**Rappresenta l'intestazione di una sezione e raggruppa uno o più elementi h1, h2, h3, h4, h5 e h6**
`<figure>`	**Rappresenta un blocco distinto dal testo principale, pensato per contenere immagini, diagrammi, esempi, etc.**
`<figcaption>`	**Rappresenta la didascalia per un elemento `figure` ed è opzionale**
`<footer>`	**Rappresenta un blocco di chiusura all'interno della pagina o di una sezione (elementi `section`, `article`, `aside` e nav)**
`<nav>`	**Rappresenta una sezione che contiene una serie di link che permettono di accedere ad altre pagine o ad altre sezioni della pagina corrente**
`<section>`	**Rappresenta una sezione generica della pagina, senza una connotazione specifica**

Possiamo usare gli elementi descritti nella tabella sopra in combinazione tra di loro, al fine di formare una pagina contenente diverse tipologie di sezioni e blocchi di markup.

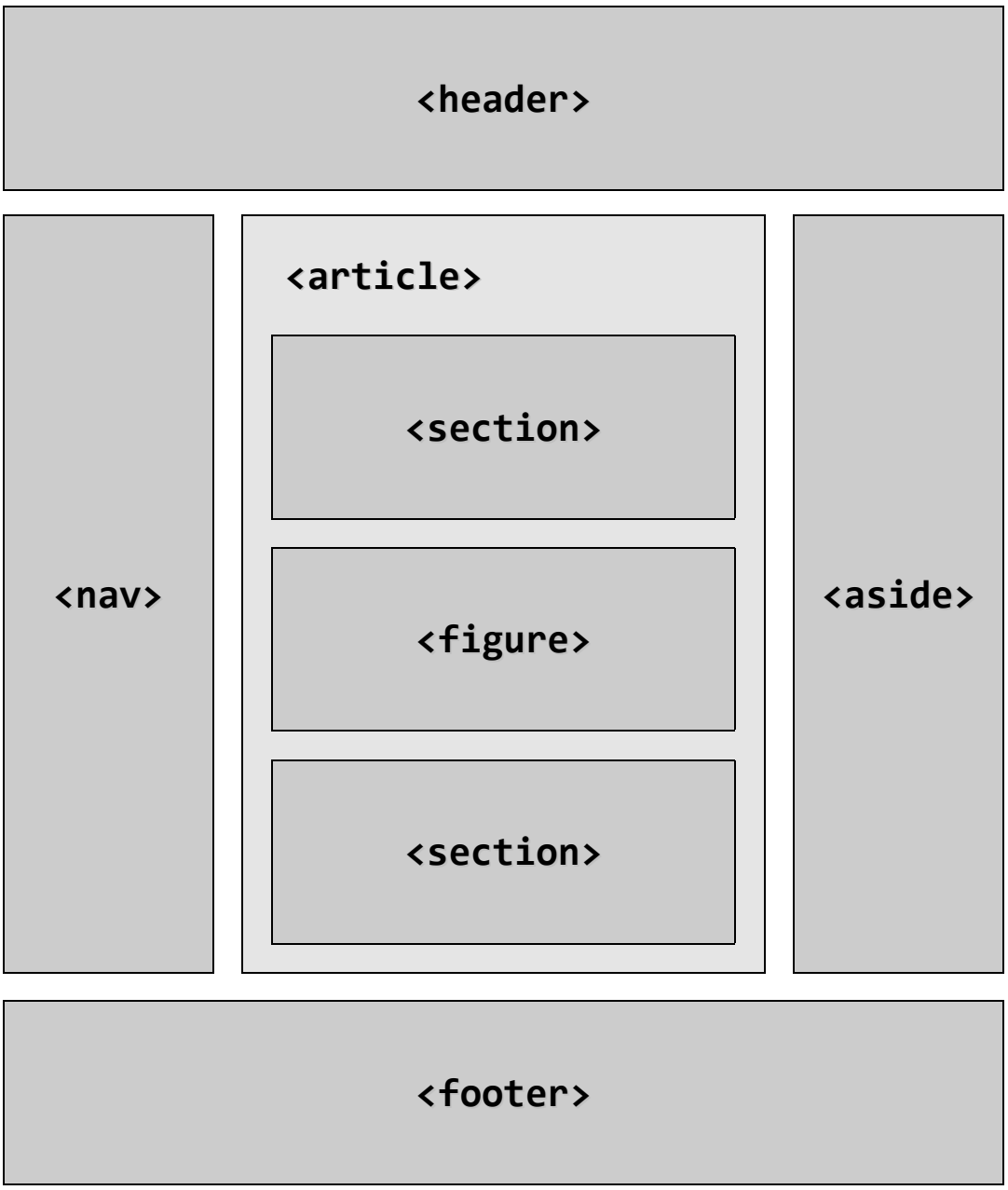
L'esempio mostra un semplice caso di formattazione di una pagina con HTML5.

```
<!DOCTYPE html>
<html>
<head>
  <title>Capitolo 2 - Nuovi elementi del markup</title>
  <link rel="stylesheet" href="../css/style.css" type="text/css" />
</head>
<body>
  <header>
    <hgroup>
      <h1>CAPITOLO 2</h1>
      <h2>Nuovi elementi del markup</h2>
    </hgroup>
  </header>
  <article>
    <section>
      <p>Sebbene le sue specifiche non siano...</p>
    </section>
    <section>
      <h3>Gli elementi supportati da HTML5</h3>
      <p>HTML5 include più di un centinaio di elementi di markup...</p>
    </section>
    <section>
      <h3>I nuovi elementi di formattazione della pagina</h3>
      <p>Man mano che le esigenze di pubblicazione dei...</p>
    </section>
    <aside>
      <h3>Fonti</h3>
      <ul>
        <li>World Wide Web Consortium</li>
        <li>Web Hypertext Application Technology Working Group</li>
      </ul>
    </aside>
  </article>
  <footer>
    <p>
      HTML5 Espresso<br />
      D. Bochicchio, C. Civera, M. Casati, R. Golia, S. Mostarda<br />
      Copyright &copy; 2011 HOEPLI
    </p>
  </footer>
</body>
</html>
```

Per avere un'idea più concreta di come una pagina formattata utilizzando i tag semantici di HTML5 possa essere strutturata all'interno di un browser, prendiamo in considerazione la figura sotto. In essa abbiamo rappresentato titolo esemplificativo una situazione di distribuzione dei contenuti abbastanza comune all'interno della pagina.

All'intestazione in alto (header) segue un corpo centrale suddiviso in tre parti (menu di navigazione sulla sinistra, area principale al centro e colonna laterale a destra), seguito a sua volta da un blocco di chiusura in basso (footer).

Figura x – Classica struttura di una pagina con i tag semantici di HTML5



Come possiamo notare, i nuovi tag di HTML5 ci permettono di rappresentare in modo mirato le diverse parti di cui la pagina si compone, aiutandola a raggiungere il miglior significato a livello semantico.

Inoltre essi favoriscono la "lettura" del markup sia da parte dei browser in grado di supportare la nuova versione del linguaggio, sia da parte di sistemi automatici come, per esempio, un bot vocale o il crawler di un motore di ricerca. In HTML5, infatti, la struttura della pagina è meno anonima e il markup diventa più descrittivo, dato che ogni elemento denota il proprio contenuto in modo chiaro, a tutto vantaggio dei motori di ricerca in grado di interpretare e indicizzare al meglio il testo in funzione dei tag presenti nel markup.

<header>

Funzioni e dati tecnici

Il tag **header** serve a rappresentare "un gruppo di ausili *introduttivi* o di navigazione". Tale definizione, seppure apparentemente vaga, contiene in sé i concetti chiave per comprendere appieno la funzione di questo tag:

1. L'elemento <header> è un contenitore per altri elementi.
2. L'elemento <header> non va confuso con quella che è la testata/intestazione principale di un documento (quella che oggi si definisce in genere con il tag <h1>).
3. La natura e gli scopi dell'elemento <header> non dipendono dalla sua posizione nel documento, ma dai suoi contenuti (ausili alla navigazione o elementi *introduttivi*).
4. Il suo uso non è obbligatorio e in alcuni casi può risultare superfluo se non utilizzato in maniera appropriata.

```
<header>
   <h1>Questo è un titolo</h1>
   <h2>Questo è un sotto-titolo</h2>
   [...]
</header>
```

Esempi concreti

Riprendendo il nostro progetto guida, dove nella lezione precedente abbiamo definito il contenuto dell'<head>:

```
<head>
   <meta charset="utf-8">
   <title> We5! Il blog della guida HTML5 </title>
   <link rel="stylesheet" href="monitor.css" media="screen">
   <link rel="stylesheet" href="printer.css" media="print">
   <link rel="stylesheet" href="phone_landscape.css"
         media="screen and (max-device-width: 480px) and
                (orientation: landscape)">
   <link rel="stylesheet" href="phone_portrait.css"
         media="screen and (max-device-width: 480px) and
                (orientation: portrait)">
   <link rel="icon" href="standard.gif" sizes="16x16" type="image/gif">
   <link rel="apple-touch-icon" href="iphone.png" sizes="57x57"
         type="image/png">
   <link rel="icon" href="vector.svg" sizes="any" type="image/svg+xml">
</head>
```

A questo punto possiamo iniziare a comporre il <body> del nostro documento partendo proprio con il tag **<header>**, che con l'elemento <hgroup> definisce il titolo principale del documento (del sito) e la cosiddetta tagline:

```
<header>
   <hgroup>
      <h1>We5! Il blog della guida HTML5</h1>
      <h2>Approfittiamo fin da oggi dei vantaggi delle specifiche
          HTML5!</h2>
   </hgroup>
</header>
```

Ma **header** non deve contenere necessariamente solo titoli <h*n*>! Se titolo e sottotitolo principali sono certamente elementi *introduttivi* ai contenuti successivi, è naturalmente un ausilio di navigazione una lista di link che andrà a formare la barra di navigazione principale del sito.

Ecco come possiamo completare la struttura del nostro primo <header>:

```
<header>
   <hgroup>
      <h1>We5! Il blog della guida HTML5</h1>
      <h2>Approfittiamo fin da oggi dei vantaggi delle specifiche
          HTML5!</h2>
   </hgroup>
   <nav>
      <h1>Navigazione:</h1>
      <ul>
         <li><a href="/home">Home</a></li>
         <li><a href="/about">Gli autori</a></li>
         <li><a href="/refactoring">Il progetto four2five</a></li>
         <li><a href="/archives">Archivio</a></li>
      </ul>
   </nav>
</header>
```

Nel seguente schema abbiamo realizzato graficamente ciò che il codice semanticamente rappresenta nel nostro progetto:

Figura 8 – Struttura del documento: primo header

Abbiamo inserito una sezione di navigazione (<nav> </nav>) introdotta da un elemento <h1> e strutturata con una lista non ordinata.

In realtà, il menu di navigazione non deve essere necessariamente inserito nell'<header>, nel nostro esempio non poteva essere fatto altrimenti ma esistono numerosi tipi di layout in cui il menu di navigazione può essere facilmente slegato dagli elementi introduttivi di intestazione del documento.

Il template del nostro progetto guida è relativo ad un blog. Nel corpo del documento, ad un certo punto, trova posto una sezione che ospita i contenuti principali della pagina, i post. Per definire semanticamente la sezione useremo il tag <section>; ciascun post sarà definito a livello strutturale da un tag <article>:

```
<section>
   <h1>L'ultimo post</h1>
   <article>
      [...]
   </article>
</section>
```

Si noti, innanzitutto, come il tag <h1> che fa da titolo principale alla sezione non sia racchiuso in un elemento **<header>**. Ribadiamo: non è obbligatorio inserire i titoli <h*n*> all'interno di un contenitore <header>.

A questo punto, dobbiamo definire due elementi fondamentali per la struttura di un post di blog: il titolo e la data. Sono certamente ausili *introduttivi*, secondo la definizione da cui siamo partiti. é più che legittimo e sensato, pertanto, racchiuderli in un tag **<header>**:

```
<section>
   <h1>L'ultimo post</h1>
   <article>
      <strong><header>
         <time datetime="2010-11-22" pubdate>Lunedì 22 Novembre</time>
         <h2>Nuove scoperte sul tag video!</h2>
      </header></strong>
      <p>
         [...]
      </p>
      <footer>
         [...]
      </footer>
   </article>
</section>
```

Ecco quindi come il nostro articolo potrebbe essere rappresentato graficamente:

Figura 9 – Struttura del documento: header degli articoli

I due scenari mostrati rendono bene l'idea rispetto agli utilizzi tipici dell'elemento `<header>`. La specifica suggerisce come altri contesti d'uso la tavola dei contenuti di una sezione, un form di ricerca o i loghi più rilevanti presenti nel documento.

<footer>

Funzioni e dati tecnici

L'elemento **<footer>** deve contenere in genere **informazioni sulla sezione che lo contiene** come:

- i dati di chi ha scritto i contenuti;
- collegamenti ai documenti correlati;
- i dati di copyright;
- e così via…

Riguardo il suo apporto semantico ad una pagina web sembra essere tutto chiaro, ma più complesso è il suo utilizzo a livello pratico:

- Non necessariamente deve essere inserito solo alla fine di un documento.

- Quando contiene intere sezioni, esse rappresentano: appendici, indici, note, accordi di licenza e altri contenuti del genere.
- Non introduce una nuova sezione e quindi non è rilevante per l'outliner.
- All'interno di una pagina web possono essere presenti diversi `<footer>` anche più di uno per lo stesso elemento.

```
<footer>
   <small>©2011 Autore contenuto. Design by Designer sito </small>
</footer>
```

Esempi concreti

Riprendendo il nostro progetto guida, dopo aver definito definito il contenuto dell'`<header>`, possiamo vedere come il **<footer>** chiuda il blog distaccandosi dalle altre sezioni in modo molto naturale, racchiudendo al proprio interno una lista che aggiunge informazioni riguardo l'autore della pagina e la data del suo ultimo aggiornamento; infine il tag `<small>` ridefinito nella specifica dell'HTML 5 racchiude il copyright della pagina:

```
<footer>
   <dl>
      <dt>Creato da</dt>
      <dd><address><a href="mailto:sandro.paganotti@gmail.com">
                           Sandro Paganotti</a></address></dd>
      <dt>Ultimo aggiornamento</dt>
      <dd>
         <time datetime="2010-11-23" pubdate>Martedì 23 Novembre</time>
      </dd>
      <dd>
   </dl>
   <small>
      Tutti i contenuti sono protetti dalla licenza creative commons
   </small>
</footer>
```

Nel seguente schema abbiamo realizzato graficamente ciò che il codice semanticamente rappresenta nel nostro progetto (figura 1):

Figura 10 – Struttura del documento: il footer principale

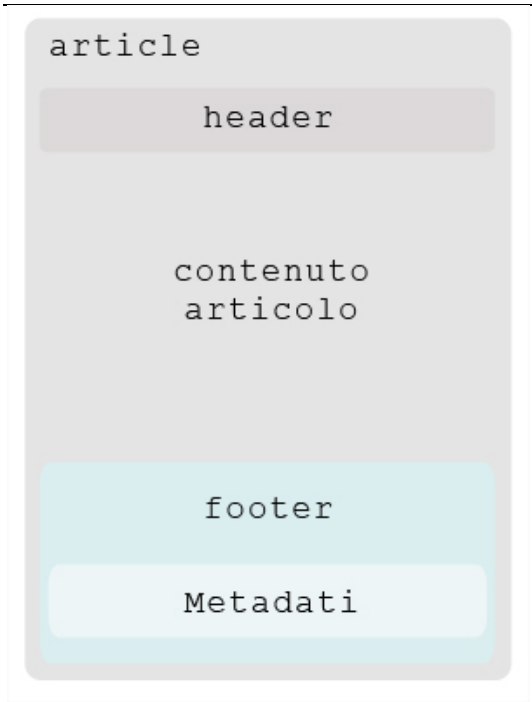
Così come l'intero documento, ogni articolo del nostro blog avrà un <footer> contenente il nome dell'autore ed altre eventuali informazioni aggiuntive:

```
<section>
   <h1>L'ultimo post</h1>
   <article>
      <header>
         [...]
      </header>
      <p>
         [...]
      </p>
      <footer>
         <dl>
            <dt>autore:</dt>
            <dd><address><a href="mailto:sandro.paganotti@gmail.com">
                                 Sandro Paganotti</a></address></dd>
            <dt>categoria: </dt>
            <dd><a href="categoria/multimedia">multimedia</a>,</dd>
            <dt>tags:      </dt>
            <dd><a href="tags/video">video</a>,</dd>
            <dd><a href="tags/canvas">canvas</a>,</dd>
            <dt>permalink: </dt>
            <dd>
               <a href="2010/22/11/nuove-scoperte-sul-tag-video">
                  permalink</a>,</dd>
            <dt>rank:</dt>
            <dd><meter value="3.0" min="0.0" max="5.0" optimum="5.0">
                  ranked 3/5</meter></dd>
         </dl>
      </footer>
   </article>
</section>
```

È da notare la scelta di inserire le informazioni riguardanti l'autore dell'articolo all'interno del tag <dl>; infatti nella specifica HTML5 questo elemento viene ridefinito come contenitore di metadati e quindi semanticamente corretto all'interno del nostro **<footer>**.

Ecco quindi come il nostro articolo potrebbe essere rappresentato graficamente, tutte le informazioni contenute nel <footer> per comodità abbiamo deciso di chiamarle metadati:

Figura 11 – Struttura del documento: footer degli articoli



L'elemento <footer> potrebbe essere inserito anche all'inizio di un documento subito dopo il <body> oppure all'apertura di un tag <article> ma in questi casi non dovrebbe contenere elementi introduttivi riguardo il contenuto della sezione che lo contiene; il suo uso in questa posizione potrebbe essere dovuto solamente a ragioni pratiche come ad esempio la duplicazione del <footer> in fondo alla pagina quando i contenuti della stessa sono molto lunghi:

```
<body>
<footer>
   <a href="#indice">Torna all'indice</a>
</footer>
<section>
   [Contenuti molto lunghi...]
<section>
<section>
   [Contenuti molto lunghi...]
<section>
<section>
   [Contenuti molto lunghi...]
<section>
<footer>
   <a href="#indice">Torna all'indice</a>
</footer>
</body>
```

<section>

Funzioni e dati tecnici

Il tag **<section>**, secondo la definizione presente nella specifica HTML5, rappresenta una **sezione generica di un documento o applicazione**. L'elemento <section>, in questo contesto, individua un raggruppamento tematico di contenuti,ed in genere contiene un titolo introduttivo.

Vediamo quindi quali sono i punti fondamentali da ricordare riguardo il suo utilizzo:

1. l'elemento <section> **non deve** essere utilizzato in sostituzione del per impostare graficamente la pagina; inoltre è fortemente consigliato utilizzare i anche quando risultano più convenienti per gli script;
2. l'elemento <section> **non deve** essere preferito all'elemento <article> quando i contenuti possono essere ripubblicati anche su altri siti;

3. l'elemento `<section>` e l'elemento `<article>` non sono indipendenti ed esclusivi: possiamo avere sia un `<article>` all interno di un `<section>` che viceversa.

```
<article>
   <section>
      <h1>Titolo 1</h1>
      <p>Testo correlato al titolo 1.</p>
   </section>
   <section>
      <h1>Titolo 2</h1>
      <p>Testo correlato al titolo 2.</p>
   </section>
</article>
```

L'elemento **`<section>`** può essere utilizzato in combinazione con l'attributo `cite` attraverso il quale è possibile specificare l'url dalla quale si stanno riportando i contenuti.

Esempi concreti

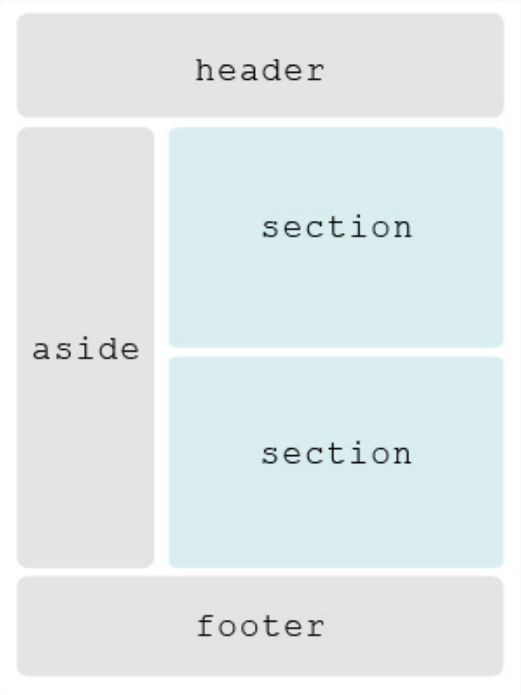
Come abbiamo visto nei capitoli precedenti, il codice del nostro progetto inizia a prendere una forma più chiara e definita: infatti, dopo aver compreso l'utilità semantica dell'`<header>` e del `<footer>`, capiamo come utilizzare l'elemento **`<section>`** all'interno del nostro blog.

Per strutturare la pagina raggruppando i contenuti correlati, in ordine decrescente incontriamo le prime due grandi macrosezioni del blog: "l'ultimo post" e "i post meno recenti" contenuti quindi in due `<section>`:

```
<section>
   <h1>L'ultimo post</h1>
   <article>
      [contenuto del post...]
      <section>
         [commenti...]
      </section>
   </article>
</section>
<section>
   <h1>Post meno recenti</h1>
   <article>
      [contenuto del post...]
   </article>
   <article>
      [contenuto del post...]
   </article>
   <article>
      [contenuto del post...]
   </article>
</section>
```

Nel seguente schema abbiamo realizzato graficamente ciò che il codice semanticamente rappresenta nel nostro progetto:

Figura 12 – Struttura del documento: sezioni principali



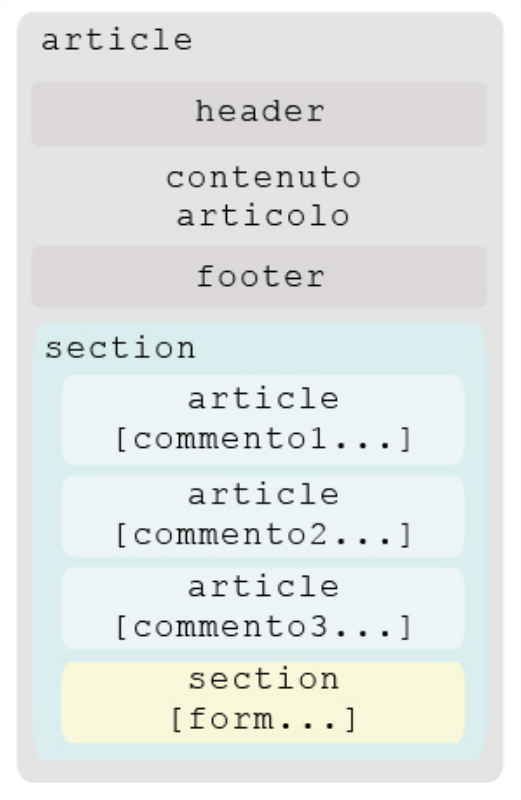
Nel nostro progetto le `<section>`, oltre a poter raggruppare i vari `<article>`, sono presenti anche all'interno del primo `<article>` per suddividere i commenti dal contenuto del post. La sezione dei commenti a sua volta contiene un'altra sezione contenente il form per l'inserimento di un nuovo commento:

```
<article>
   [contenuto del post...]
   <section>
      <article>
         [commento1...]
      </article>
      <article>
         [commento2...]
      </article>
      <article>
         [commento3...]
      </article>
      <section>
         [Inserisci un nuovo commento...]
      </section>
   </section>
</article>
```

In questo modo il post è diviso in maniera molto netta rispetto ai propri contenuti solo con l'ausilio dei tag HTML5, separando quindi i commenti che sono una sezione aggiuntiva eventualmente anche eliminabile dall'argomento principale trattato all'interno dell'articolo.

Lo schema dell'articolo analizzato è quindi il seguente:

Figura 13 – Struttura del documento: suddivisione semantica del post



Il tag **<section>** rappresenta un elemento che viene considerato una sezione a sé stante dall'outliner e quindi un blocco con dei contenuti univoci che necessitano di un titolo (<hN>) che li riassuma. Come vedremo, esistono anche altri elementi nelle specifiche HTML5 che sono considerati "contenitori di sezionamento dei contenuti".

<article>

Funzioni e dati tecnici

Il tag **<article>** rappresenta una **sezione autonoma in un documento, pagina, applicazione o sito**; infatti è potenzialmente ridistribuibile o riutilizzabile, e quindi ripubblicabile in parte o interamente in diverse pagine.

Esso può identificare il post di un forum, un articolo di una rivista o di un giornale, l'articolo di un blog, un commento, un widget interattivo, o qualsiasi cosa che abbia un contenuto indipendente.

Prima di vedere un esempio concreto bisogna chiarire alcuni concetti riguardo il suo utilizzo:

1. quando gli elementi `<article>` sono nidificati, gli `<article>` interni rappresentano gli articoli che sono in linea di principio relativi al contenuto dell'`<article>` esterno. Ad esempio, un blog che accetta commenti dagli utenti potrebbe rappresentarli come `<article>` figli annidati all'interno dell'elemento padre `<article>`;
2. le informazioni relative all'autore dell'`<article>` non devono essere replicate all'interno degli elementi nidificati all'interno dello stesso;
3. l'elemento `<time>` con l'attributo `pubdate` può essere utilizzato per definire la data di pubblicazione dell'`<article>`;
4. l'elemento `<section>` e l'elemento `<article>` non sono indipendenti ed esclusivi: possiamo avere sia un `<article>` all interno di un `<section>` che viceversa.

```
<article>
   <header>
      <h1>Titolo articolo</h1>
      <time pubdate datetime="2011-10-09T14:28-08:00"></time></p>
   </header>
   <p>Contenuto dell'articolo</p>
   <footer>
      <p>Informazioni riguardo l'autore</p>
   </footer>
</article>
```

In sostanza, anche se nelle specifiche non è espresso, l'elemento `<article>` rappresenta una forma particolare di `<section>`.

Esempi concreti

Nella lezione precedente abbiamo diviso i contenuti centrali del blog che stiamo costruendo in due `<section>`. All'interno della prima `<section>` possiamo trovare diversi tag **`<article>`**: il primo che incontriamo è l'articolo vero e proprio con tanto di `<header>` contenente il titolo dell'articolo e la data di pubblicazione e `<footer>` che all'interno di un `<dl>` raccoglie i metadati riguardanti l'autore.

All'interno dell'`<article>` padre sono annidati ulteriori `<article>` contenenti i commenti all'articolo racchiusi in una `<section>` che li rende semanticamente separati dal contenuto principale. Così come i commenti, anche il form che permette di inserire un'ulteriore commento è inserito all'interno di una `<section>`. Possiamo quindi facilmente immaginare come il contenuto del nostro `<article>` possa essere citato o ripubblicato in altri blog indipendentemente dai commenti che ha ricevuto.

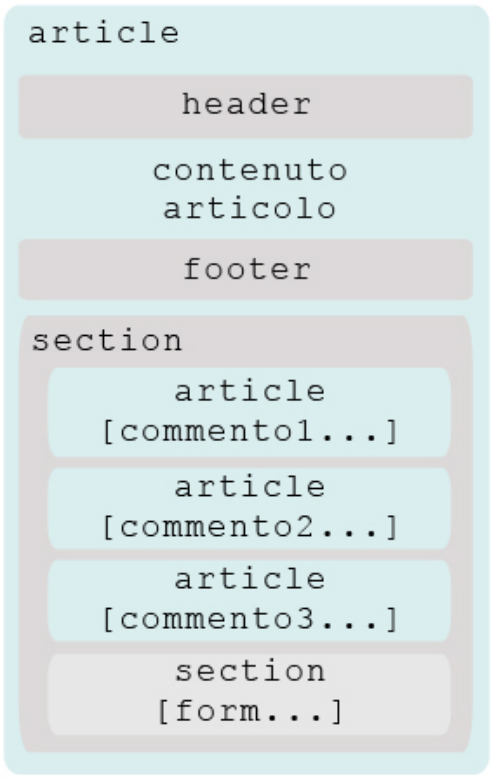
Ecco quindi il codice dell'`<article>` sopra descritto:

```
<section>
   <h1>L'ultimo post</h1><article>
   <header>
      <time datetime="2010-11-22" pubdate>Lunedì 22 Novembre</time>
      <h2>Nuove scoperte sul tag video!</h2>
   </header>
   <p>
      Attraverso un utilizzo sapiente del tag canvas
      è possibile leggere uno stream
      di dati proveniente da un tag video
      e <mark>manipolarlo in tempo reale</mark>.
   </p>
   <footer>
      <dl>
         <dt>autore:    </dt>
         <dd><address><a href="mailto:sandro.paganotti@gmail.com">
                        Sandro Paganotti</a></address></dd>
         <dt>categoria: </dt>
         <dd><a href="categoria/multimedia">multimedia</a>,</dd>
         <dt>tags:      </dt>
         <dd><a href="tags/video">video</a>,</dd>
         <dd><a href="tags/canvas">canvas</a>,</dd>
         <dt>permalink: </dt>
         <dd><a href="2010/22/11/nuove-scoperte-sul-tag-
                     video">permalink</a>,</dd>
         <dt>rank:</dt>
         <dd><meter value="3.0" min="0.0" max="5.0" optimum="5.0">
             ranked 3/5</meter></dd>
      </dl>
   </footer>
   <section>
```

```
      <h3>Commenti</h3>
      <article>
         <h4>
            <time datetime="2010-11-22" pubdate>
            Lunedì 22 Novembre</time>Angelo Imbelli ha scritto:
         </h4>
         <p>C'è un bell'esempio sulla rete: effetto ambi-light!</p>
         <footer>
            <address><a href="mailto:ambelli@mbell.it">
                     Angelo Imbelli</a></address>
         </footer>
      </article>
      <article>
         <h4>
            <time datetime="2010-11-23" pubdate>Martedì 23 Novembre
            </time>
            Sandro Paganotti ha scritto:
         </h4>
         <p>Bellissimo! Grazie per la segnalazione!</p>
         <footer>
            <address><a href="mailto:sandro.paganotti@gmail.com">
                     Sandro Paganotti</a>
            </address>
         </footer>
      </article>
      <section>
         <h4>Inserisci un nuovo commento:</h4>
         <form>
            [ campi form per inserire un nuovo commento]
         </form>
      </section>
   </section>
</section>
```

Nel seguente schema abbiamo realizzato graficamente come si presenta il nostro articolo morfologicamente:

Figura 14 – Struttura del documento: suddivisione semantica degli articoli



C'è da notare che anche se il tag **<article>** rappresenta un elemento che viene considerato una sezione a sé stante dall'outliner e quindi un blocco con dei contenuti unici e un titolo univoco (quindi per ogni blocco avremmo potuto utilizzare un titolo racchiuso in un <h1>), abbiamo preferito rispettare comunque l'ordine decrescente per i titoli in modo da rendere i contenuti accessibili anche agli screen reader che al momento non hanno ancora implementato l'algoritmo outliner.

<nav>

Funzioni e dati tecnici

Il tag **<nav>** è uno degli elementi introdotti nelle specifiche HTML5 di più facile comprensione. Infatti, rappresenta una sezione di una pagina che **contiene link (collegamenti) ad altre pagine o a parti interne dello stesso documento**; quindi, in breve, una sezione contenente **link di navigazione**.

A questo punto potremmo porci una domanda: come mai un elemento così scontatamente fondamentale è stato introdotto solamente adesso? La risposta potrebbe essere che, così come per i tag visti nelle precedenti lezioni, finalmente si è cercato di incentivare l'uso di standard condivisi proponendo elementi che possano aiutare gli sviluppatori proprio perché molto vicini ai modelli mentali oramai assimilati dagli esperti e di semplice comprensione per i novizi del mestiere.

Per poter utilizzare correttamente l'elemento <nav> dobbiamo ricordare i seguenti punti:

1. solo sezioni che sono costituite da **grandi blocchi di navigazione** sono appropriati per l'elemento <nav>;
2. i link a pie' di pagina e le breadcumb **non devono** essere inseriti in una sezione <nav>;
3. l'elemento <nav> **non sostituisce** i link inseriti in elementi come gli <ul> o gli <ol> ma deve racchiuderli.

```
<nav>
   <ul>
      <li>Questo è un link</li>
      <li>Questo è un link</li>
      [...]
   </ul>
</nav>
```

Esempi concreti

Prima di spiegare in che modo l'elemento **<nav>** può essere inserito nel progetto che abbiamo preso come base, riassumiamo brevemente i tag spiegati nelle lezioni precedenti:

- Con l'elemento <header> abbiamo indicato il titolo introduttivo del blog più i titoli dei singoli articoli.
- Con il <footer> abbiamo racchiuso le informazioni relative agli autori dei contenuti, i metadati e il copyright.
- Con l'elemento <section> abbiamo strutturato la parte centrale della pagina dividendola in due sezioni semanticamente distinte.
- Infine abbiamo utilizzato <article> per racchiudere i contenuti degli articoli.

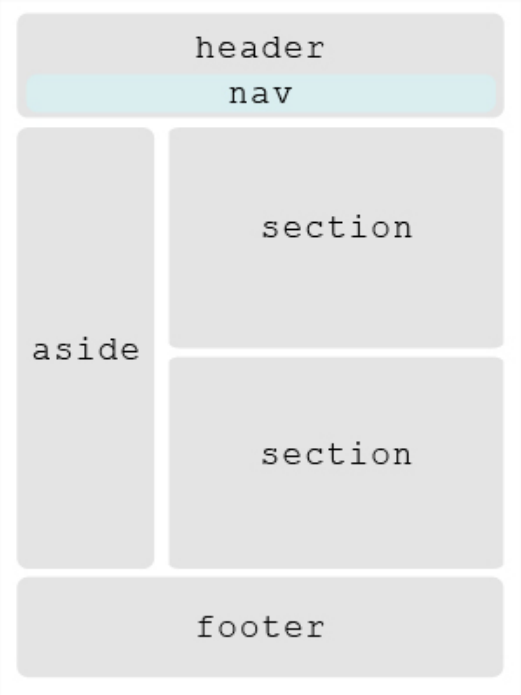
Adesso non possiamo che inserire i link presenti nell'<header> all'interno del tag **<nav>**:

```
<header>
   <hgroup>
      <h1>We5! Il blog della guida HTML5</h1>
      <h2>Approfittiamo fin da oggi dei vantaggi delle specifiche HTML5!</h2>
   </hgroup>
   <strong><nav>
      <h1>Navigazione:</h1>
      <ul>
         <li><a href="/home">Home</a></li>
         <li><a href="/about">Gli autori</a></li>
         <li><a href="/refactoring">Il progetto four2five</a></li>
         <li><a href="/archives">Archivio</a></li>
      </ul>
   </nav></strong>
</header>
```

Da notare la presenza del titolo <h1> che serve a specificare più dettagliatamente la funzione del nostro blocco di link e a conferirgli un titolo valido anche per l'outliner. Il tag **<nav>**, infatti, rappresenta un elemento che viene considerato una sezione a sé stante dall'outliner e quindi un blocco con dei contenuti univoci che necessitano di un titolo che li riassuma.

Nel seguente schema abbiamo realizzato graficamente ciò che il codice semanticamente rappresenta nel nostro progetto:

Figura 15 – Struttura del documento: visualizzazione grafica del tag nav



In realtà (come già specificato nel paragrafo <header>) il menu di navigazione **non deve essere necessariamente inserito nel** <header>, nel nostro esempio non poteva essere fatto altrimenti ma esistono numerosi tipi di layout in cui il menu di navigazione può essere facilmente slegato dagli elementi introduttivi di intestazione del documento.

Nel nostro esempio l'elemento <nav> è presente anche nella colonna laterale del blog (<aside>) e racchiude un menu che ha come link le categorie nelle quali sono inseriti i vari articoli:

```
<aside>
   <h1>Sidebar</h1>
   <section>
      <h2>Ricerca nel form:</h2>
         <form>
            [form di ricerca dei contenuti...]
         </form>
   </section>
   <strong><nav>
      <h2>Categorie</h2>
      <ul>
         <li><a href="/categoria/multimedia">multimedia</a></li>
         <li><a href="/categoria/text">marcatori testuali</a></li>
         <li><a href="/categoria/form">forms</a></li>
      </ul>
   </nav></strong>
</aside>
```

<aside>

Funzioni e dati tecnici

L'elemento **<aside>** rappresenta una sezione di una pagina costituita da **informazioni che sono marginalmente correlate al contenuto dell'elemento padre che la contiene**, e che potrebbero essere considerate distinte da quest'ultimo. Questo è ciò che viene indicato nelle specifiche HTML5, ma è facile immaginare l'utilità del tag <aside> semplicemente pensandolo come un **contenitore di approfondimento** in cui possiamo inserire gruppi di link, pubblicità, bookmark e così via.

```
<aside>
   <h1>Questi sono dei contenuti di approfondimento marginali
       rispetto al contenuto principale</h1>
   <nav>
      <h2>Link a pagine correlate al contenuto</h2>
      <ul>
         <li>Informazione correlata al contenuto</li>
         <li>Informazione correlata al contenuto</li>
         <li>Informazione correlata al contenuto</li>
      </ul>
   </nav>
   <section>
      <h2>Pubblicità</h2>
      [immagini pubblicitarie]
   <section>
</aside>
```

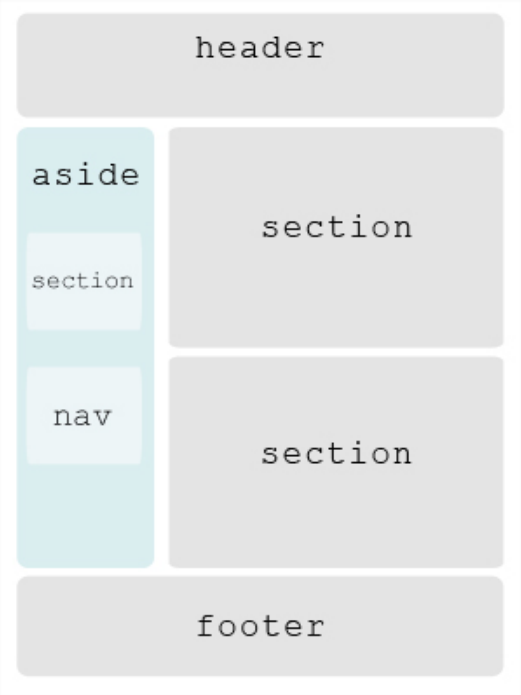
Aside: esempi concreti

Ritornando al nostro progetto guida, dopo aver definito il contenuto dell'elemento <nav>, possiamo analizzare la parte di codice in cui abbiamo utilizzato il tag **<aside>**:

```
<aside>
   <h1>Sidebar</h1>
   <section>
      <h2>Ricerca nel form:</h2>
      <form name="ricerca" method="post" action="/search">
         <label> Parola chiave:
            <input type="search" autocomplete="on" placeholder="article,
            section, ..." name="keyword" required maxlength="50">
         </label>
         <input type="submit" value="ricerca">
      </form>
   </section>
   <nav>
      <h2>Categorie</h2>
      <ul>
         <li><a href="/categoria/multimedia">multimedia</a></li>
         <li><a href="/categoria/text">marcatori testuali</a></li>
         <li><a href="/categoria/form">forms</a></li>
      </ul>
   </nav>
</aside>
```

Nel seguente schema abbiamo realizzato graficamente ciò che il codice semanticamente rappresenta nel nostro progetto:

Figura 16 – Struttura del documento: visualizzazione grafica del tag aside



Come possiamo notare, il form per la ricerca di parole chiave e i link alle categorie presenti nel nostro blog non sono informazioni particolarmente rilevanti per il contenuto centrale della nostra pagina, pertanto possiamo facilmente separarli con l'elemento `<aside>` che li qualifica come contenuti marginali.

Nel nostro caso abbiamo utilizzato un `<aside>` per contenere la colonna sinistra del blog, ma in realtà, visto che in diversi siti va di moda la presenza di footer molto grossi con diversi link, consigliamo di utilizzare l'elemento **`<aside>`** in combinazione con il tag `<nav>` che potrebbe essere la soluzione migliore per questa tipologia di contenuti dato che, come abbiamo potuto constatare nella lezione dedicata al footer, ciò non è possibile farlo all'interno del tag `<footer>`.

Così come gli elementi `<article>`, `<section>` e `<nav>` anche l'`<aside>` appartiene alla categoria dei "contenitori di sezionamento dei contenuti" in quanto per l'outliner necessita di un titolo che riassuma i propri contenuti.

Ricordiamo però che non è obbligatorio inserire un titolo per gli elementi che vengono considerati delle sezioni a sé stanti dall'outliner, infatti in questo caso queste sezioni rimarrebbero senza titolo ma non genererebbero nessun errore di validazione.

Vediamo quindi ora che abbiamo completato la struttura del blog come è l'outline del nostro documento Outline.

`<hgroup>`

Funzioni e dati tecnici

L'elemento `<hgroup>` rappresenta l'intestazione di una sezione. L'elemento viene utilizzato per raggruppare un insieme di elementi `h1-h6`, quando il titolo ha più livelli, come sottotitoli, titoli alternativi o slogan.

```
<hgroup>
   <h1>Questo è il titolo</h1>
   <h2>Questo è un sottotitolo</h2>
</hgroup>
```

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.

Esempio:

```
<article datetime="2010-11-22" pubdate >
   <header>
      <strong><hgroup>
         <h2>Le prospettive per il futuro del web</h2>
         <h1>L'HTML 5 rivoluzionerà il mondo del web</h1>
      </hgroup></strong>
   </header>
   <p>Presto il web sarà pieno di siti e applicazioni che saranno
      in grado di mettere in crisi le più grandi case di desktop
     application...</p>
   <footer>
      <p>Pinco pallino</p>
   </footer>
</article>
```

Se facessimo l'outline della pagina HTML contenente questo <article> ci restituirebbe come titolo della sezione solamente l'<h1> contenuto nell'<hgroup> (non considerando l'<h2> anche se posto prima nel codice) poiché è il tag con il valore più alto all'interno della sezione. Senza l'elemento <hgroup> i due titoli verrebbero considerati entrambi sullo stesso livello d'importanza come titoli dell' <article>.

<mark>

Funzioni e dati tecnici

L'elemento **<mark>** rappresenta una parte di un testo segnato o evidenziato all'utente a causa della sua rilevanza anche in un altri contesti. Esso, in un testo in prosa, indica un punto culminante che non era originariamente presente, ma che è stato aggiunto per attirare l'attenzione del lettore su una parte del testo che potrebbe non essere stata considerata rilevante dall'autore originale quando il contenuto è stato scritto.

Questa definizione abbastanza complessa in realtà può essere semplificata di molto chiarendoci le idee con un esempio: immaginiamo di cercare una determinata parola chiave in diverse pagine web e che la parola che abbiamo cercato nel motore di ricerca, ci venga evidenziata all'interno della pagina che andiamo a visualizzare; ciò che abbiamo appena descritto è la situazione ideale per l'utilizzo del tag **<mark>** poiché con quest'ultimo dobbiamo racchiudere la parola ricercata segnalandola graficamente all'utente.

```
<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>
```

Allo stato attuale non esiste un tag HTML standard utilizzato per evidenziare le parole chiave agli utenti che utilizzano i motori di ricerca per navigare: Google utilizza il tag , Bing il tag , Yahoo il tag <b> e così via. Si spera che con l'introduzione dell'elemento <mark> le cose possano cambiare.

<time>

Funzioni e dati tecnici

L'elemento **<time>** rappresenta **il tempo su un orologio di 24 ore**, o una **data precisa nel calendario Gregoriano** accompagnata opzionalmente con un orario e una differenza di fuso orario.

Questo elemento è inteso come un modo moderno per codificare le date e gli orari in maniera leggibile anche per i computer. Ad esempio, i browser saranno in grado di offrire la possibilità di aggiungere promemoria per i compleanni o gli eventi in programma in una web application che funziona da calendario.

```
<p>Oggi pomeriggio penso che sarò lì per le <time>15:00</time></p>
```

Prima di inserire l'elemento **<time>** nelle nostre pagine in HTML5 dobbiamo comprendere quali sono i contesti in cui è sconsigliato utilizzarlo:

- **non bisogna** inserire nel tag <time> le date che non possono essere determinate con precisione; ad esempio: "un giorno nel lontano inverno del '68","da quando è nato il primo uomo"…;
- **non bisogna** inserire nel tag <time> le date prima dell'introduzione del calendario Gregoriano.

L'elemento **<time>** può possedere l'attributo **pubdate** che è di tipo booleano; la sua presenza indica che la data presente nel tag <time> è anche la data nella quale è stato scritto l'<article> padre più vicino, e nel caso non esistesse un <article> padre allora essa è riferita alla creazione dei contenuti dell'intero documento.

Ovviamente un elemento che possiede l'attributo pubdate **necessita di una data**. Per ciascun <article> può esserci solo un singolo tag <time> con pubdate e la stessa cosa vale per l'intero documento.

Possiamo specificare in maniera più dettagliata una data aggiungendo l'attributo **datetime**:

- il valore dell'attributo deve essere una "stringa valida" del tipo (ANNO-MESE-GIORNO-ORE:MINUTI:SECONDI.MILLISECONDI-FUSO ORARIO).
- se l'attributo datetime non è presente allora il contenuto testuale del'tag <time> deve essere una "stringa valida".

```
<time pubdate datetime="2011-01-20">20 Gennaio</time>
```

Dobbiamo specificare che l'attributo pubdate in quanto di tipo booleano può essere inserito anche nel seguente modo:

```
<time pubdate="pubdate" datetime="2011-01-20">20 Gennaio</time>
```

<meter>

Funzioni e dati tecnici

L'elemento **<meter>** rappresenta una **misura scalare all'interno di un intervallo noto**, o un **valore frazionario**.

Il tag **<meter>** è anche utilizzato come un indicatore di livello.

```
<meter value="6" max="8">6 blocchi utilizzati (su un totale di 8)
</meter>
```

Vediamo alcuni contesti in cui non deve essere utilizzato:

- quando indica lo stato di una barra di progresso;
- quando i valori rappresentati sono di tipo arbitrario a scalare; ad esempio non deve segnalare un peso o un'altezza a meno che non vi sia un valore massimo riconosciuto.

Esistono 6 attributi per determinare il valore semantico dell'elemento **<meter>**:

1. min: indica il valore limite minimo disponibile;
2. max: indica il valore limite massimo disponibile;
3. value: indica il valore dell'elemento;
4. low: indica un valore che viene considerato basso;
5. high: indica un valore che viene considerato alto;

6. `optimum`: indica un valore che viene considerato "ottimale"; se è più alto del valore specificato nell'attributo `high` indica che il valore più alto è il migliore,viceversa è più basso del valore specificato nell'attributo `low` indica che il valore più basso è il migliore.

Se non è specificato un valore minimo e un valore massimo questi sono di default rispettivamente 0 e 1.

Ad oggi l'unico browser che renderizza il tag **<meter>** è Google Chrome presentandolo graficamente come una barra di progresso, quindi gli sviluppatori sono fortemente incoraggiati a specificare il suo valore in formato testuale al suo interno.

<progress>

Funzioni e dati tecnici

L'elemento **<progress>** rappresenta **lo stato di completamento di un compito** e può essere di due tipi:

- **indeterminato**: indica che il compito (task) è in fase di completamento, ma che non è chiaro quanto ancora resta da fare prima che l'operazione sia completata (ad esempio perché l'operazione è in attesa della risposta di un host remoto);
- **determinato** quando è possibile ricavare un numero quantificabile nel range da zero a un massimo, in modo da ottenere la percentuale di lavoro completato rispetto al totale(ad esempio un preloader di un'immagine).

Esistono due attributi che determinano lo stato di completamento dell'attivita del tag `<progress>`. L'attributo `value`, che specifica la quantità di lavoro completata, e l'attributo `max`, che specifica la quantità di lavoro richiesta in totale. Le unità sono arbitrarie e non specificate.

Tuttavia, i valori passati come attributi del tag **<progress>** non sono renderizzati e quindi dovrebbero comunque essere inseriti in forma testuale nell'HTML in modo da poter fornire un feedback più preciso agli utenti; inoltre questo elemento attualmente non viene renderizzato dalla maggior parte dei browser ed è quindi ancora **sconsigliato il suo utilizzo**.

```
<section>
   <p>Caricamento: <progress id="mioLoader" max="100" value="30">
                   <span>30</span>%</progress></p>
</section>
```

Questo elemento funziona nelle più recenti versioni di Opera e Google Chrome.

Gli attributi `value` e `max`, se presenti, devono essere valori validi. L'attributo `value`, se presente, deve avere un valore uguale o maggiore di zero e minore o uguale al valore dell'attributo `max`, se presente, o 1.0, in caso contrario. L'attributo `max`, se presente, deve avere un valore maggiore di zero.

Ovviamente, l'elemento **<progress>** deve essere utilizzato solamente per indicare lo stato in fase di progressione di un compito; per indicare quantitativamente la misura di un'oggetto o di uno stato non in progressione bisogna utilizzare l'elemento `<meter>`.

Data la complessità dell'argomento e la costante variazione delle specifiche a riguardo consigliamo di consultare il sito del WHATWG per un maggiore approfondimento.

<figure>, <figcaption>

Nell'elemento **<figure>** possiamo racchiudere dei contenuti, opzionalmente con una didascalia (**<figcaption>**), che rappresentano delle singole unità indipendenti rispetto al contenuto principale; ad esempio possiamo utilizzare l'elemento `<figure>` per annotare le illustrazioni, schemi, foto, elenchi di codice, etc… ovvero tutto ciò che fa parte del contenuto principale ma che potrebbe essere anche allontanato dallo stesso senza intaccarne il senso.

L'elemento `<figcaption>` quindi rappresenta una didascalia o una leggenda per l'elemento `<figure>` padre.

Esempio

```
<figure>
   <img src="benevenuti.jpg" alt="">
   <figcaption>
      Foto di benvenuto
      <small>© Diritti riservati</small>
   </figcaption>
</figure>
```

È importante notare che l'attributo `alt` è vuoto poiché l'immagine è descritta nel tag `<figcaption>` ed è stato usato il tag `<small>` per il copyright.

<embed>

L'elemento **<embed>** è già utilizzato da anni per inserire nel codice HTML contenuti interattivi o multimediali (tipicamente Flash, Quicktime, etc.). Questo elemento, però, era stato deprecato nelle specifiche HTML 4 in favore del tag `<object>`. Ora è stato reintrodotto perché, nonostante la potenza delle nuove API HTML5, si pensa che al momento non tutto ciò che si riesce ad ottenere con plug-in di terze parti possa essere realizzato in HTML5. Inoltre, si è cercato di mantenere la retrocompatibilità con applicazioni basate sull'utilizzo di questo elemento.

<ruby>

Il tag **<ruby>** è usato per specificare le annotazioni Ruby, che vengono utilizzate nella tipografia orientale in combinazione con il testo principale.

<wbr>

Il tag **<wbr>** definisce dove, in una parola, sarebbe opportuno aggiungere un a capo. Infatti, quando una parola è lunga, utilizzando l'elemento `<wbr>` il browser comprenderà dove eventualmente sarà possibile inserire un a capo.

<command>, <menu>

Entrambi sono elementi molto interessanti: permettono di definire barre degli strumenti o menu di scelta rapida per le nostre applicazioni, con le icone e i relativi comandi che possono essere eseguiti da script.

Il tag **<command>** rappresenta un'azione che l'utente può richiamare in qualche modo. Esso è visibile solo se inserito all'interno di un elemento **<menu>**. In caso contrario, non verrà visualizzato, ma può essere utilizzato per specificare un tasto di scelta rapida.

Al momento nessun browser supporta questi tag.

<details>, <summary>

I tag **<details>** e **<summary>** rappresentano un widget informativo da cui l'utente può ottenere informazioni supplementari o controlli aggiuntivi. Nel tag `<summary>`, che è contenuto all interno del tag `<details>`, deve essere inserita una sintesi del contenuto del widget o anche una legenda. I contenuti dell'elemento `<details>` possono essere mostrati o meno dal browser grazie all'attributo `open`, di tipo booleano. Anche questi tag non sono supportati ancora da nessun browser.

<keygen>

L'elemento **<keygen>** rappresenta un generatore di chiavi numeriche all'interno di un form. Quando si effettua l'invio di un form contenente il tag `<keygen>`, la chiave privata viene memorizzata nel keystore locale e la chiave pubblica viene confezionato e inviata al server.

L'elemento è già supportato da diversi browser ma manca il suo supporto in IE.

<output>

L'elemento **<output>** ci restituisce il risultato di un calcolo.