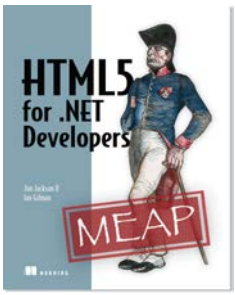
MANNING PUBLICATIONS



HTML5 for .NET Developers
By Jim Jackson II and Ian Gilman

HTML is, very simply, the tags that surround content delivered to you in your browser. It has gone through several iterations, the latest being HTML 5. This article based on chapter 2 of HTML5 for .NET Developers explains how semantic HTML can make your content more accessible and versatile.

You may also be interested in…

The Semantic Blueprint

Semantic HTML refers to the use of block-level and inline HTML 5 tags that describe what kind of content they contain. It provides two major values to an HTML page. First, it allows search engines to easily determine what content should be indexed and what content is just *page noise* used for styling or navigation. Second, it allows you, as the page developer, to segregate your page in a much more maintainable way. When you come back to a semantic page in a year or two to do some tweak, it will be easier to find your way around if the page has its blueprint built right in. Finally, it makes your content more accessible by allowing screen-reading tools to discern what is important content on your page and what parts are just navigational noise.

We have divided the HTML 5 elements up into three categories. Content tags are wrappers for content while application tags are used to present information about how your program is operating. Media tags are used to present rich content.

What is the point of semantic HTML?

Think of it this way. In my house we have three external doors. If I wanted a package delivered, I could tell the delivery company to drop it off at *door 2* but that tells the driver nothing. I could also say to drop off the package at the door marked *side* but he is still going to have to search. Now, if you send a diagram with each door marked *front*, *side*, and *back* and tell him to use the side door, he will know ahead of time where he is going and how to get there. Describing your page semantically is the same concept.

<section> (Content)

`<section>` defines content that should be treated as a document within the page. Based on the HTML 5 specification, it should not be used as a styling or scripting hook—it should be used to denote a specific piece of content. Bruce Lawson wrote at HTML 5doctor.com, "Don't use it unless there is naturally a heading at the start of the section." Using the section as a styling mechanism does make sense though when looking at the larger semantic scheme. Why would you locate and style elements based on an article but not a section? My opinion is to style away with sections; just be sure to keep them semantically appropriate.

```
<section>
   <p>Age considers; youth ventures</p>
</section>
```

<header> (Content)

Earlier HTML specifications required the use of only one `<h1>` tag in an HTML page. With the section and article comes the ability to organize multiple content targets in a page and the `<header>` element is the means by which that target can have its own head elements (`<h1>`, `<h2>`, etc.).

```
<section>
```

```
   <header>
      <h1>Famous Quotes</h1>
      <h2>Victor Hugo on Age</h2>
   </header>
   <p>Forty is the old age of youth; fifty is the youth of old age.</p>
</section>
```

<footer> (Content)

The <footer> performs the same function as <header> in a piece of target content—only it goes at the end. It provides a means of segregating the *ending* of the target content.

```
<section>
   <p>Fortune and love favor the brave.</p>
   <footer>Ovid</footer>
</section>
```

<article> (Content)

<article> is often confused with <section> because they both perform a page-level segregation of content. They can also have <header> and <footer> elements. The big difference between an article and a section is their level of independence. While a <section> is designed to be a part of its parent element or page, the <article> is designed to be an independently distributable piece of content.

```
<article>
   <h1>March scheduled activities</h1>
   <p>Come in like a lion</p>
   <p>Go out like a lamb</p>
</article>
<article>
   <h1>April schedule activities</h1>
   <p>Bloomin' flowers</p>
</article>
```

This leads to additional confusion because a page can have multiple sections, each with its own articles. Likewise, a page can have multiple articles with various sections inside each article. In the former case, the articles are related to the section but not to each other. In the latter case, the sections are natural dividing points within each article. Semantically speaking, this makes sense.

```
<section>
   <header>
      <h1>Recipies</h1>
   </header>
   <article>
      <h1>30 minute apple pie</h1>
      <p>...</p>
   </article>
   <article>
      <header>
         <h1>Herb-braised ham with white wine</h1>
      </header>
      <p>...</p>
   </article>
</section>
```

or

```
<article>
   <header>
      <h1>Quotables</h1>
   </header>
   <section>
      <header>
         <h1>Parenting</h1>
      </header>
      <p>Kids are educated by who the parent is, not what he says.</p>
   </section>
   <section>
      <header>
         <h1>Politics</h1>
      </header>
      <p>A fool and his money are soon elected.</p>
   </section>
<article>
```

A good way to think of the difference between an <article> and a <section> is to consider section as a verb and article as a noun. A page can be sectioned but the sections still comprise the page. Alternatively, a page can have articles but each article ought to be distributable without the page it was originally placed on.

<aside> (Content)

An <aside> tag exists inside an article or section element and allows for related information to be presented about the subject matter without disrupting the main content flow. It is a content presentation tag, not a layout tag, so avoid using it as the wrapper for sidebar-style <menu> structures.

```
<article>
   <p>All war is deception.</p>
   <aside>Sun Tzu: Chinese General and author (Art of War)</aside>
   <p>Strategy without tactics is the slowest route to victory.</p>
   <p>Tactics without strategy is the noise before defeat.</p>
</article>
```

<details> (Content)

The <details> element is interesting in that it has an open attribute. By default, this element and attribute are just grouping elements for content but, when activated using JavaScript or via built-in browser capabilities, they provide an additional function controlling whether they are displayed (open) or hidden (no attribute).

<summary> (Content)

The <summary> tag is the means of describing a title or heading for a details element. It exists inside a <details> element and, in supported browsers, provides the ability to toggle the open attribute.

```
<p>Common Auto Maintenance Tasks</p>
<details open="open">
   <summary>Scheduling Notes</summary>
   <em>Oil change schedules will vary.</em>
</details>
<ul>
   <li>Change oil</li>
   <li>Rotate tires</li>
</ul>
```

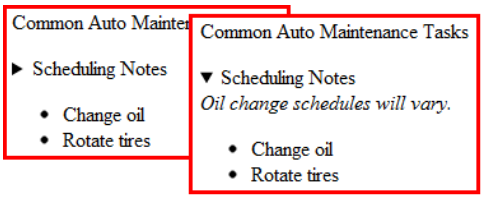


Figure 1 <details> element in both the expanded and contracted mode with <summary> providing clickable heading (rendered in Chrome)

<figure> (Content)

The <figure> tag is used the same way as figures in books, magazines, and technical papers. An image or illustration that provides value to the content is placed within an <article> or <section> to provide additional contextual value.

```
<figure>
   <img src="blueridgemtns.jpg" />
</figure>
```

<figcaption> (Content)

The <figcaption> tag is contained within a <figure> element and provides the caption content for that <figure>. Because a single <figure> element can contain many source images, the <figcaption> can refer to one or more elements inside its parent figure.

```
<figure>
   <img src="skylinedrive.jpg" />
   <figcaption>105 mile road through Shenandoah Natioanl Park</figcaption>
```

```
</figure>
```

<hgroup> (Content)

The <hgroup> is technically used to group one or more <h1>, <h2>, and so on and provide only the top level as part of the document outline. This is generally only useful for text-rich web sites and CMS systems. Note also that <hgroup> may be removed entirely from the HTML 5 spec, so use it sparingly, if at all.

<mark> (Content)

<mark> tags surround text in the page content that is relevant to the user. The most common use of the <mark> tag is to highlight search results in a piece of content. <mark> is different from in that it provides notation of user relevance, not just contextual emphasis by the original author.

```
<p><mark>Wimbledon</mark> is a district in the south west area of London, England, located
south of Wandsworth, and east of Kingston upon Thames. It is situated within Greater London.
It is home to the <mark>Wimbledon<mark> Tennis Championships and New <mark>Wimbledon<mark>
Theatre, and contains <mark>Wimbledon<mark> Common.</p>
```

<nav> (Content)

The <nav> tag is a wrapper for the primary navigation elements on your page. <nav> can contain any other element; it is simply a means of grouping the important parts of your page or site navigation in one place. Not every link or menu structure needs to be located in a <nav> but it is helpful for what you consider the important navigation paths to be contained in a nav.

```
<nav>
   <ul>
      <li><a>Home</a></li>
      <li><a>Back</a></li>
      <li><a>Next</a></li>
   </ul>
</nav>
```

<canvas> (Application)

<canvas> is both a tag and a complete drawing API. You can draw pretty much anything you like in a <canvas> as well as style the element itself using normal properties. <canvas> can be used to build images on the fly or to allow your users the ability to incorporate existing graphics into their experience. This is a deep topic and I recommend that, if you want to learn more, you check out any of the numerous blogs and books on the subject. The tag looks simple enough but it is the entrance point to a significant API.

```
<canvas />
```

<command> (Application)

The <command> tag, when placed inside a <menu> element, will define a hook to execute your code. <command> elements can have a type value of checkbox, radio, or command, the last of which provides a normal button-style operation.

```
<menu>
   <command type="command">Save all documents</command>
   <command type="checkbox">Check to include current document</command>
</menu>
```

<datalist> (Application)

<datalist> is a hidden tag on your page that provides a list of values for another element. When an <input> element is assigned a list attribute, it is given the id that corresponds to the <datalist> against which its values should be validated. The validation process is your responsibility, although the browser may soon provide you with some help in this regard.

```
<input type="text" name="autos" list="mfg">
<datalist list="mfg">
   <option value="Ford" />
   <option value="Toyota" />
   <option value="Ferrari" />
</datalist>
```

<keygen> (Application)

<keygen> is a major step toward client-initiated authentication in HTML. When implemented by the browser, this element will generate a public/private key pair and send the public key to the server with the enclosing form. At the time of this writing, <keygen> is incomplete, but keep it in mind that, when it is finalized, it will be an important part of your security architecture.

```
<keygen name="myformKey" form="newUserInput" />
```

<progress> (Application)

The <progress> tag gives you a quick way to show percentage of progress for any task. It contains only two required attributes, value and max. The minimum value is implied at zero, which makes this tag unsuitable for displaying measurement values or scalar values in a range. The <meter> tag would be more appropriate for that.

```
<progress max="120" value="90"></progress>
```

<meter> (Application)

The <meter> tag is similar to the <progress> in presentation and function but it has the ability to show an amount in a scale that starts from a non-zero value. This is done by filling in the min and max attributes for the range and the value attribute.

```
<meter min="25" max="75" value="60"></meter>
```

<time> (Application)

The <time> tag is not a means of formatting a time value but rather a means of stating a piece of content is a date, time, or date/time for purposes of semantic clarity. The <time> tag can be used to contain a date/time value and it can also be used to enhance the contents of a date/time value. This can be done with the datetime attribute.

```
<p>The Gettysburg Address, given on <time>November 19, 1863</time> started with the words
"<time datetime="July 1776">Four score and seven years ago our fathers brought
forth...</time></p>
```

<source> (Media)

A <source> tag is used to define a piece of media content within a media element. It contains a src attribute as well as a type attribute that corresponds to its media type. (Media types are the correct nomenclature for what are commonly referred to as mime types.)

```
<source src="Chrome.mp3" type="audio/mpeg"></source>
```

<audio> (Media)

The <audio> tag represents audio content on a page. A src attribute can be set or you can use multiple <source> elements inside along with the old-school <object> tags for injecting plugin players when the browser does not support the formats specified in the source elements. Static or streaming audio content can be used depending on how your server is configured.

```
<audio>
   <source src="CountryBoy.mp3" type="audio/mpeg" />
   <source src="CountryBoy.ogg" type="audio/ogg" />
   <object>Silverlight player here</object>
   Your browser does not support any available audio format.
</audio>
```

or

```
<audio src="DownOnTheCorner.mp3" type= "audio/mpeg" />
```

<video> (Media)

The <video> tag performs the same task for video as the <audio> tag does for audio content. The same caveats apply relative to video formats and server configuration. The src attribute is also present, as is the ability to nest <source> and <object> elements inside a <video> element. The video element has some added features like

`loop` to make the video automatically restart and `poster` to describe the image that should appear before the video starts. `autoplay` is also available.

```
<video controls="controls" autoplay="autoplay" poster="StartImg.jpg">
   <source src="DemolitionMan.mp4" type="video/mp4" />
   <source src="DemolitionMan.ogg" type="video/ogg" />
   <object></object>
</video>
```

Summary

In this article, we raced through a lot of new and a few old-school features around HTML as an application framework. We divided HTML elements into three categories and showed you how to blend the new with the old to do some really amazing things for the new generation web content.

Here are some other Manning titles you might be interested in:

Quick & Easy HTML5 and CSS3
Rob Crowther

Sass and Compass in Action
Wynn Netherland, Nathan Weizenbaum, and Chris Eppstein

Secrets of the JavaScript Ninja
John Resig and Bear Bibeault

Last updated: February 1, 2012