A user-friendly reference guide

# HTML5 & CSS3

SAMPLE CHAPTER

Rob Crowther

MANNING

# Hello! HTML5 & CSS3

by Rob Crowther

## Chapter 1

# Brief contents

Part 1

# Learning HTML5

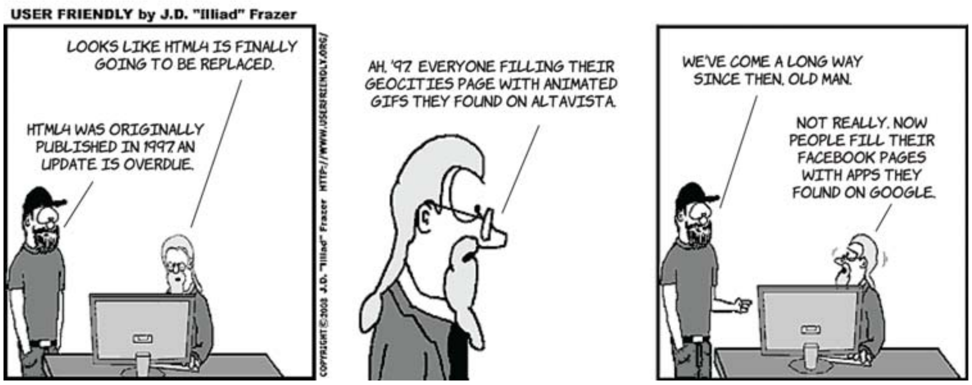
This part of the book focuses on HTML5. Chapter 1 introduces you to new and updated markup features in HTML5, chapter 2 discusses forms and form validation, chapter 3 explores HTML5's new dynamic graphics capabilities, chapter 4 talks about how to use video and audio on your web pages, and chapters 5 and 6 look at the new APIs you can use for client-side development and networking.

1

# Introducing HTML5 markup

## This chapter covers

- *New semantic elements in HTML5*
- *Updated HTML4 elements*
- *New global attributes*
- *The HTML5 content model*
- *Getting new elements to work in old browsers*


USER FRIENDLY by J.D. "Illiad" Frazer
LOOKS LIKE HTML4 IS FINALLY GOING TO BE REPLACED.
HTML4 WAS ORIGINALLY PUBLISHED IN 1997. AN UPDATE IS OVERDUE.
COPYRIGHT ©2008 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
AH, '97. EVERYONE FILLING THEIR GEOCITIES PAGE WITH ANIMATED GIFS THEY FOUND ON ALTAVISTA.
WE'VE COME A LONG WAY SINCE THEN, OLD MAN.
NOT REALLY. NOW PEOPLE FILL THEIR FACEBOOK PAGES WITH APPS THEY FOUND ON GOOGLE.

This chapter assumes you have some knowledge of previous versions of HTML. If you're new to HTML, check out appendix B—it should give you enough information to understand this chapter.

We'll start with some background on how and why the particular set of new elements in HTML5 was chosen. Then we'll examine new elements for the overall structure of web pages before moving on to elements, both new and redefined, intended for particular bits of content. You'll then learn about the new attributes in HTML5. Next, we'll spend a few pages considering the more conceptual issue of the new approach to element categorization in HTML5. Finally, you'll go back to practicalities and learn how to make sure your new HTML5 content will work in old browsers.

## Why do we need new elements?

This section looks at some of the research that went into understanding the document structures that web authors were trying to describe semantically with HTML; this information was used to decide which new elements should be added in HTML5. We'll then look at each of the new elements in turn.

### What does *semantic* mean?

At heart, HTML is a way of describing hyperlinked documents: documents that are linked together as part of a network of knowledge. The elements of HTML are meant to *mean something*, and that meaning is what we refer to as the *semantics*. Because HTML describes documents, the semantics are along the lines of "this content is a paragraph," "this content is a level-one heading," and "this content is an unordered list."

Being able to describe the structure of a document this way is valuable because it lets you keep the details of how to best display content separate from the content itself. The result is that the same web page, if well structured, can easily be read on a desktop computer, a mobile phone, and a text-to-speech converter. Compare this to a document format like PDF, where the layout and content are deeply interlinked because the fidelity of the eventual printed output is the primary goal. It's usually awkward to read an A4 PDF on a mobile device because there's no option other than to view it at A4 size.

HTML4 has two built-in methods for extending the semantics of elements: the `id` and `class` attributes. The `id` attribute is a unique identifier, but, rather than a random string, the identifier can be a meaningful word—in other words, it can have semantic value. The class isn't unique, but multiple classes can be applied to a single element like tagging in popular social network tools. Some examples are shown in the following table.

| Markup | Suggested meaning |
|---|---|
| `<p>` | A paragraph |
| `<p id="author">` | A paragraph that represents a particular author |
| `<p class="bio">` | A paragraph that represents a biography |
| `<p class="author bio">` | A paragraph that represents an author biography |

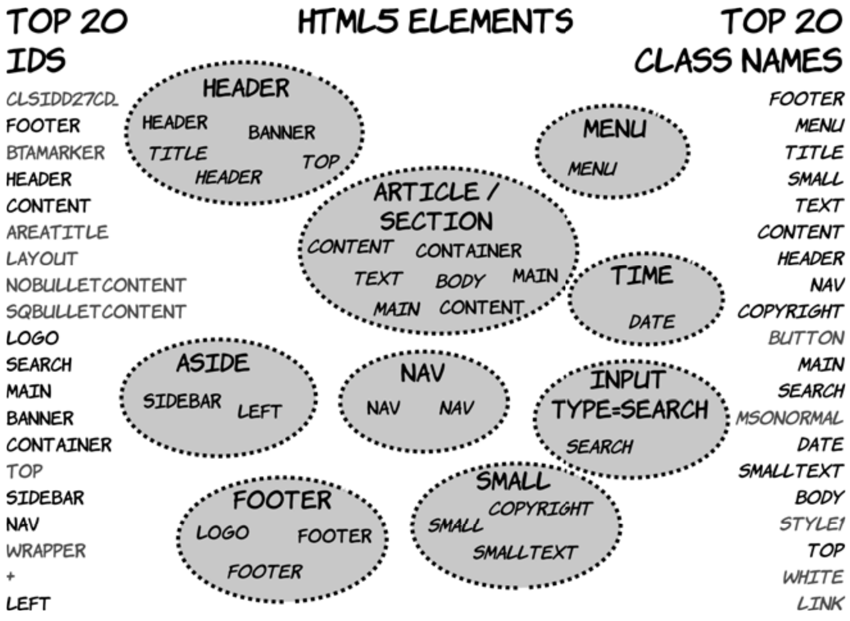
No definitive standard sets down which values mean what,[1] so one site could use *writer* for the same thing another site uses *author* for, or two sites could use *author* to mean something completely different. This isn't a huge issue, because HTML isn't intended to describe real-world things like authors, so the meaning behind those values is likely to be site-specific anyway. But `id` and `class` attributes can also be used to describe document features; for instance, a `nav` class would probably indicate an element that contains navigation. If you were looking for ideas for new elements to add to HTML to improve its ability to describe documents, a survey of the sorts of values used in `id` and `class` attributes would be a good place to start.

With this in mind, in 2005 several studies were done that attempted to analyze how authors were using `id` and `class` values in markup on the web. Two of these are of particular interest to us:

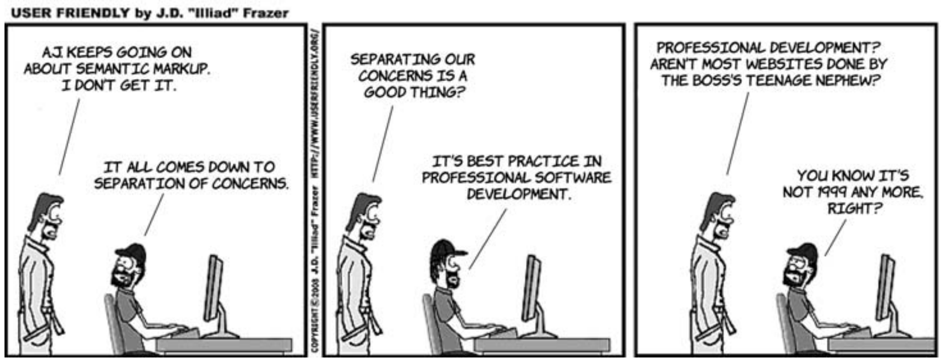
- In November 2005, a study of 1,315 websites counted how often different values for the `id` attribute were used.
- In December 2005, a study of slightly over a billion web pages analyzed, among other things, how often particular class names appeared.

1 Although some have attempted it. See the discussion of microformats later in this chapter.

The diagram that follows shows the top 20 results in each category down each side and the corresponding new HTML5 elements along with the IDs and classes that inspired them in the middle.



MANY OF THE TOP IDS, LIKE `btamarker` AND `nobulletcontent`, ARE AUTOMATICALLY GENERATED BY SOFTWARE SUCH AS MICROSOFT FRONTPAGE AND OTHER OFFICE PRODUCTS. THEIR POPULARITY IS THEREFORE MORE AN INDICATION OF THE MARKET PENETRATION OF THE PRODUCTS THAN AUTHOR REQUIREMENTS OR INTENTIONS.


USER FRIENDLY by J.D. "Illiad" Frazer
AJ KEEPS GOING ON ABOUT SEMANTIC MARKUP. I DON'T GET IT.
IT ALL COMES DOWN TO SEPARATION OF CONCERNS.
COPYRIGHT © 2008 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
SEPARATING OUR CONCERNS IS A GOOD THING?
IT'S BEST PRACTICE IN PROFESSIONAL SOFTWARE DEVELOPMENT.
PROFESSIONAL DEVELOPMENT? AREN'T MOST WEBSITES DONE BY THE BOSS'S TEENAGE NEPHEW?
YOU KNOW IT'S NOT 1999 ANY MORE, RIGHT?

In the next section, you'll learn about some of the new elements that have been added to HTML5 as a result of this research.

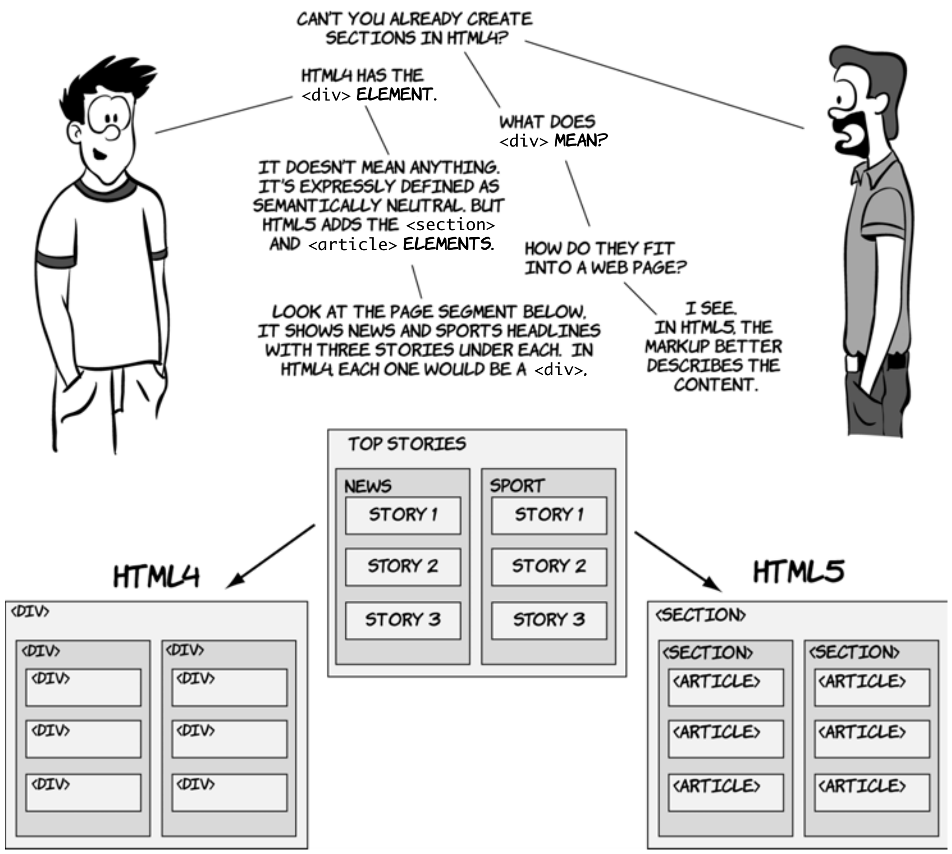
## New elements for page structure

By *page structure* we mean the top-level items: the header, the footer, the navigation, the main content, and so on. Let's join A.J. and Greg, who are discussing the research results from the previous section.

OK, WE NEED NEW ELEMENTS, AND THE RESEARCH POINTS AT WHAT THEY SHOULD BE IN GENERAL. HOW DOES THIS FIT INTO REAL WEBSITES?
WHY DON'T WE PICK SOME WEBSITES AND TAKE A LOOK TOGETHER?
I VISIT THE BBC AND WIKIPEDIA EVERY DAY.
AND I'LL ADD THE WORDPRESS BLOG TO GIVE US A GOOD CROSS SECTION.
THAT SEEMS LIKE THREE QUITE DIFFERENT SITES!
AND YET YOU'LL SEE THAT THEY HAVE MANY COMMON ELEMENTS.
Just Another WordPress Weblog

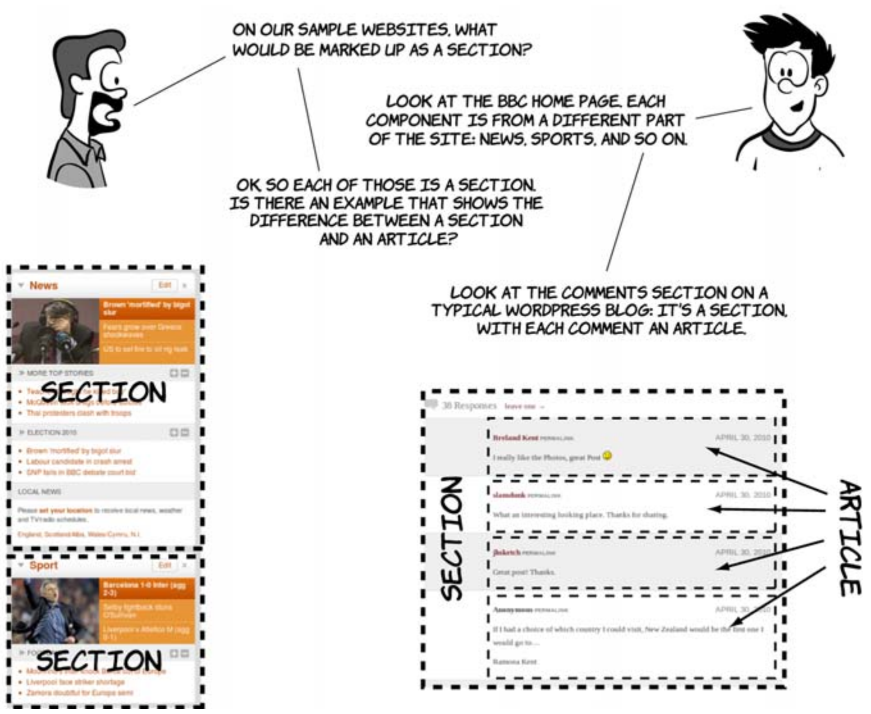
### Sectioning content

It's common for web pages to have many distinct sections. A blog homepage usually has several blog posts, each a section in itself, and each blog post may have a comments section or a related-posts section. HTML4 offers only one type of element for this common need: `<div>`. HTML5 adds two new elements: `<section>` and `<article>`.


CAN'T YOU ALREADY CREATE SECTIONS IN HTML4?
HTML4 HAS THE <div> ELEMENT.
WHAT DOES <div> MEAN?
IT DOESN'T MEAN ANYTHING. IT'S EXPRESSLY DEFINED AS SEMANTICALLY NEUTRAL. BUT HTML5 ADDS THE <section> AND <article> ELEMENTS.
HOW DO THEY FIT INTO A WEB PAGE?
LOOK AT THE PAGE SEGMENT BELOW. IT SHOWS NEWS AND SPORTS HEADLINES WITH THREE STORIES UNDER EACH. IN HTML4, EACH ONE WOULD BE A <div>.
I SEE. IN HTML5, THE MARKUP BETTER DESCRIBES THE CONTENT.
TOP STORIES
NEWS
SPORT
STORY 1
STORY 2
STORY 3
HTML4
<DIV>
HTML5
<SECTION>
<ARTICLE>

The `<section>` and `<article>` elements are conceptually similar. Articles and sections can be interchangeable—articles can exist happily within sections, but articles can also be broken down into sections, and there's been a lot of discussion about whether HTML5 really needs both of them. For now, though, we have both, and you're probably asking yourself how to decide which one to use. The key parts of the spec to focus on when choosing one or the other are as follows:

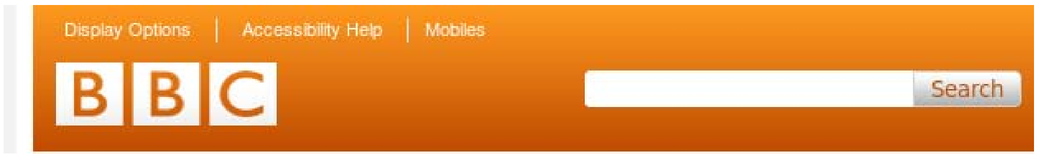
- An *article* is intended to be independently distributable or reusable.
- A *section* is a thematic grouping of content.


ON OUR SAMPLE WEBSITES, WHAT WOULD BE MARKED UP AS A SECTION?
LOOK AT THE BBC HOME PAGE. EACH COMPONENT IS FROM A DIFFERENT PART OF THE SITE: NEWS, SPORTS, AND SO ON.
OK SO EACH OF THOSE IS A SECTION. IS THERE AN EXAMPLE THAT SHOWS THE DIFFERENCE BETWEEN A SECTION AND AN ARTICLE?
LOOK AT THE COMMENTS SECTION ON A TYPICAL WORDPRESS BLOG: IT'S A SECTION, WITH EACH COMMENT AN ARTICLE.
News
SECTION
Sport
SECTION
SECTION
ARTICLE

## Headings, headers, and the outlining algorithm

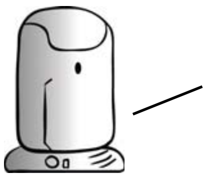
Heading elements provide an implicit structure for documents. A *heading* indicates the start of a new section and briefly describes the topic of the text that follows. The level of a heading (levels 1 through 6 in HTML) indicates an implicit hierarchy. This implicit structure is useful for the automatic generation of a table of contents. Some websites, such as Wikipedia, generate a table of contents for each page; screen readers and other accessibility tools use the table of contents to allow users to navigate the page more easily. HTML5 formalizes this implicit structure with the outlining algorithm. In this section, you'll learn about this algorithm as well as how it interacts with the two new heading elements, `<header>` and `<hgroup>`.

A `<header>` element appears near the top of a document, a section, or an article and usually contains the main heading and often some navigation and search tools. Here's an example from the BBC website.



Here's how that might be marked up in HTML5:

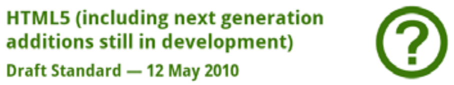
```
<header>
  <h1>BBC</h1>
  <nav>
    <ul>
      <li><a href="/options">Display Options</a></li>
      <li><a href="/access">Accessibility Help</a></li>
      <li><a href="/mobile">Mobiles</a></li>
    </ul>
  </nav>
  <form target="/search">
    <input name="q" type="search">
    <input type="submit">
  </form>
</header>
```



YOU'LL LEARN MORE ABOUT THE <nav> ELEMENT LATER IN THIS CHAPTER. HTML5'S NEW FORM ELEMENTS WILL BE COVERED IN DEPTH IN CHAPTER 2.

The <hgroup> element should be used where you want a main heading with one or more subheadings. For an example, let's look at the HTML5 spec:

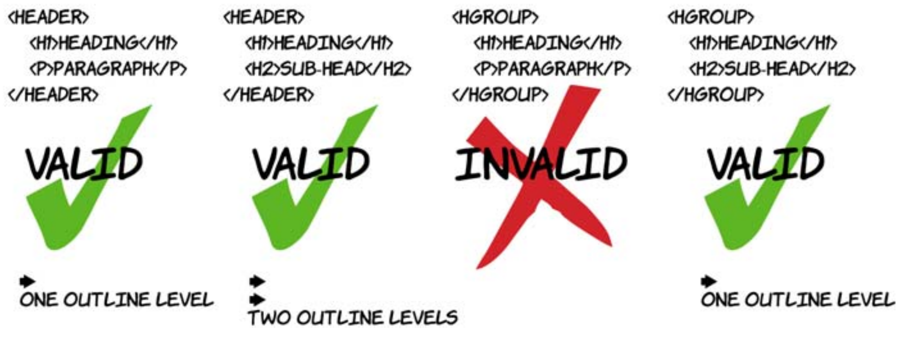
```
<hgroup>
  <h1>HTML5
    (including next generation
     additions still in
development)
   </h1>
  <h2>Draft Standard &mdash;
      12 May 2010</h2>
</hgroup>
```



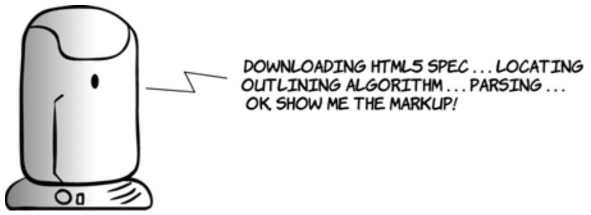
**HTML5 (including next generation additions still in development)**

**Draft Standard — 12 May 2010**

The `<header>` element can contain any content, but the `<hgroup>` element can only contain other headers—that is, `<h1>` to `<h6>`, plus `<hgroup>` itself. The following diagram demonstrates the differences.

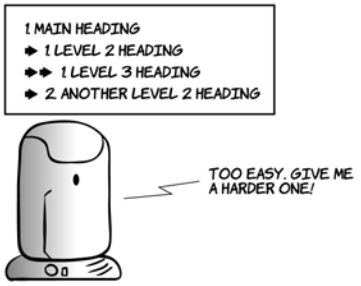


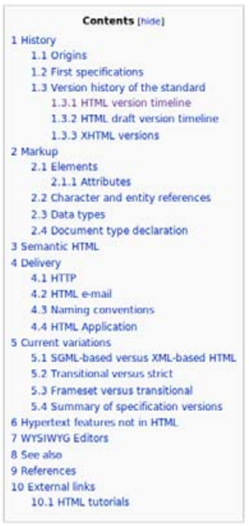
The outlining algorithm generates a table of contents for your document based on the section and heading markup you've used. In HTML4, the overall structure of a document was left up to individual browsers to decide; in HTML5, it's part of the spec. This benefits you because any user agents that need an outline, often for accessibility purposes,[2] will generate the same outline for any given document. To help you get the idea, let's look at several sample documents. Erwin will generate the document outline according to the HTML5 spec. You'll see how the outline is impacted both by headings and heading groups as well as the articles and sections we discussed in the previous section.


DOWNLOADING HTML5 SPEC . . . LOCATING OUTLINING ALGORITHM . . . PARSING . . . OK SHOW ME THE MARKUP!

[2] The W3C's User Agent Accessibility Guidelines recommend that browsers generate a document outline in guideline 1.10.2: www.w3.org/TR/UAAG20/#gl-alternative-views.

```
<body>
    <h1>Main heading</h1>
    <p>Some text</p>
    <h2>Level 2 heading</h2>
    <p>Some more text</p>
    <h3>Level 3 heading</h3>
    <p>A bit more text</p>
    <h2>Another level 2 heading</h2>
    <p>The last bit of text</p>
</body>
```



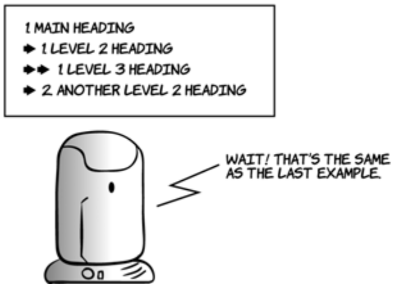


**Contents** [hide]

1 History
  1.1 Origins
  1.2 First specifications
  1.3 Version history of the standard
    1.3.1 HTML version timeline
    1.3.2 HTML draft version timeline
    1.3.3 XHTML versions
2 Markup
  2.1 Elements
    2.1.1 Attributes
  2.2 Character and entity references
  2.3 Data types
  2.4 Document type declaration
3 Semantic HTML
4 Delivery
  4.1 HTTP
  4.2 HTML e-mail
  4.3 Naming conventions
  4.4 HTML Application
5 Current variations
  5.1 SGML-based versus XML-based HTML
  5.2 Transitional versus strict
  5.3 Frameset versus transitional
  5.4 Summary of specification versions
6 Hypertext features not in HTML
7 WYSIWYG Editors
8 See also
9 References
10 External links
  10.1 HTML tutorials

In a plain document with no other sectioning content, the outline will match the heading levels. This is similar to the way a table of contents in Wikipedia is generated (right). Headings can also be grouped using the <hgroup> element. Let's see how they affect the document outline:

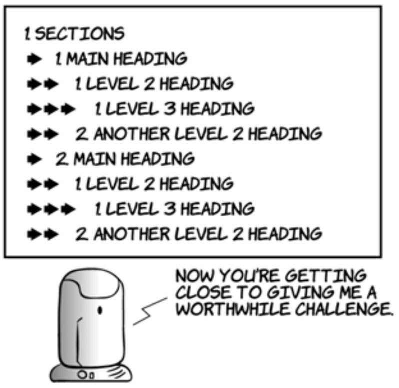
```
<hgroup>
    <h1>Main heading</h1>
    <h2>
      Subheading to main heading
    </h2>
</hgroup>
<p>Some text</p>
<h2>Level 2 heading</h2>
<p>Some more text</p>
<h3>Level 3 heading</h3>
<p>A bit more text</p>
<hgroup>
    <h2>Another level 2 heading</h2>
    <h3>
      Subheading to level 2 heading
    </h3>
</hgroup>
<p>The last bit of text</p>
```


1 MAIN HEADING
➧ 1 LEVEL 2 HEADING
➧➧ 1 LEVEL 3 HEADING
➧ 2 ANOTHER LEVEL 2 HEADING
WAIT! THAT'S THE SAME AS THE LAST EXAMPLE.

The outline will only show the highest level heading from any `<hgroup>`: you can see the headings "Subheading to main heading" and "Subheading to level 2 heading" don't appear in the outline. The `<hgroup>` element can contain any number of subheadings, but it can only contain other heading elements.

Next, let's look at how sections affect the outline:

```
<h1>Sections</h1>
<section>
    <h1>Main heading</h1>
    <p>Some text</p>
    <h2>Level 2 heading</h2>
    <p>Some more text</p>
    <h3>Level 3 heading</h3>
    <p>A bit more text</p>
    <h2>Another level 2 heading</h2>
    <p>The last bit of text</p>
</section>
<section>
    <h1>Main heading</h1>
    <p>Some text</p>
    <h2>Level 2 heading</h2>
    <p>Some more text</p>
    <h3>Level 3 heading</h3>
    <p>A bit more text</p>
    <h2>Another level 2 heading</h2>
    <p>The last bit of text</p>
</section>
```

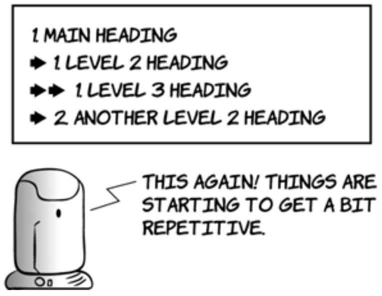


```
1 SECTIONS
➧ 1 MAIN HEADING
➧➧ 1 LEVEL 2 HEADING
➧➧➧ 1 LEVEL 3 HEADING
➧➧ 2 ANOTHER LEVEL 2 HEADING
➧ 2 MAIN HEADING
➧➧ 1 LEVEL 2 HEADING
➧➧➧ 1 LEVEL 3 HEADING
➧➧ 2 ANOTHER LEVEL 2 HEADING
```

NOW YOU'RE GETTING CLOSE TO GIVING ME A WORTHWHILE CHALLENGE.

As you can see, there are now multiple `<h1>` elements in the document, but they don't all sit at the same level of the document outline. In fact, you can do without any heading element other than `<h1>`. Let's look at another example.

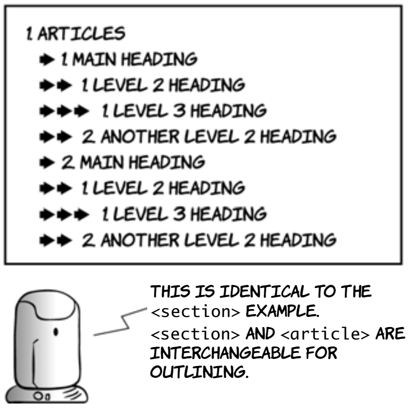
```
<h1>Main heading</h1>
<p>Some text</p>
<section>
    <h1>Level 2 heading</h1>
    <p>Some more text</p>
    <article>
        <h1>Level 3 heading</h1>
        <p>A bit more text</p>
    </article>
</section>
<section>
    <h1>Another level 2 heading</h1>
    <p>The last bit of text</p>
</section>
```



```
1 MAIN HEADING
➧ 1 LEVEL 2 HEADING
➧➧ 1 LEVEL 3 HEADING
➧ 2 ANOTHER LEVEL 2 HEADING
```

We have achieved the exact same outline as the original example but using only level-one headings. Earlier, we discussed the similarity between `<section>` and `<article>`. If we replace one with the other in the previous listing, you can see how similar they are:

```
<h1>Articles</h1>
<article>
    <h1>Main heading</h1>
    <p>Some text</p>
    <h2>Level 2 heading</h2>
    <p>Some more text</p>
    <h3>Level 3 heading</h3>
    <p>A bit more text</p>
    <h2>Another level 2 heading</h2>
    <p>The last bit of text</p>
</article>
<article>
    <h1>Main heading</h1>
    <p>Some text</p>
    <h2>Level 2 heading</h2>
    <p>Some more text</p>
    <h3>Level 3 heading</h3>
    <p>A bit more text</p>
    <h2>Another level 2 heading</h2>
    <p>The last bit of text</p>
</article>
```

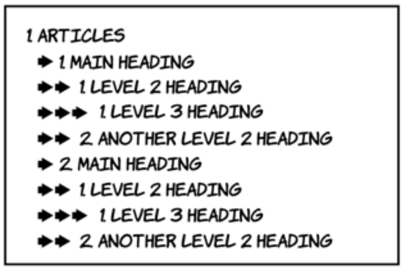


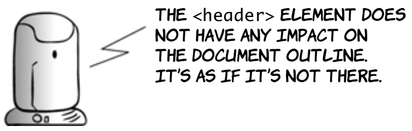
```
1 ARTICLES
  ➧ 1 MAIN HEADING
  ➧➧ 1 LEVEL 2 HEADING
  ➧➧➧ 1 LEVEL 3 HEADING
  ➧➧ 2 ANOTHER LEVEL 2 HEADING
  ➧ 2 MAIN HEADING
  ➧➧ 1 LEVEL 2 HEADING
  ➧➧➧ 1 LEVEL 3 HEADING
  ➧➧ 2 ANOTHER LEVEL 2 HEADING
```

THIS IS IDENTICAL TO THE <section> EXAMPLE.
<section> AND <article> ARE INTERCHANGEABLE FOR OUTLINING.

Now let's consider the <header> element. It represents the header of a document, a section, or an article, typically containing headings and other metadata about the section. You'll frequently have content that you don't want to be part of the heading element itself but that doesn't fit in with the following content. Examples would be subheadings, author bylines, and publishing date information:

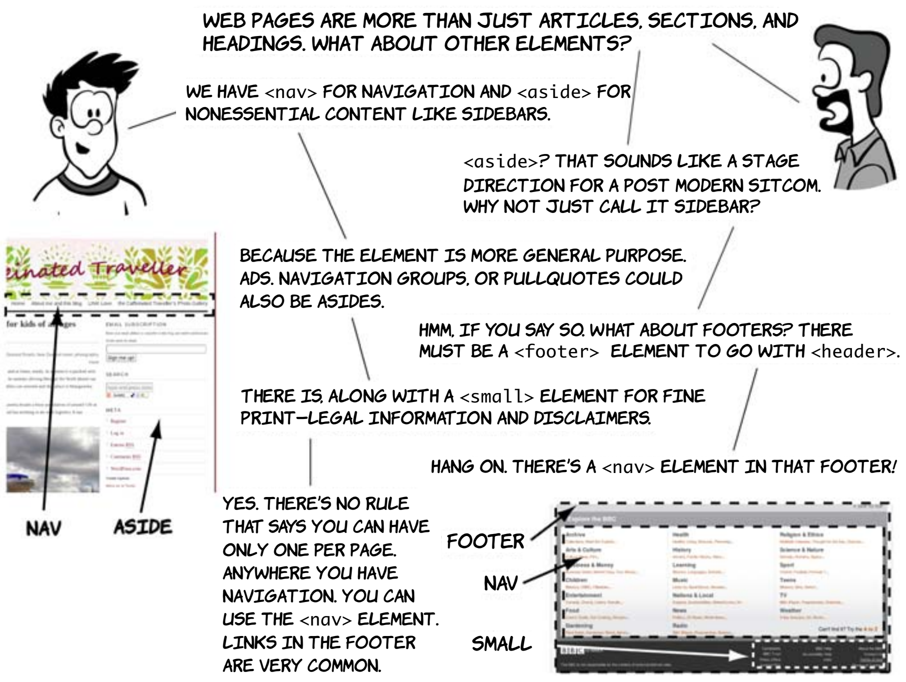
```
<h1>Articles</h1>
<article>
    <header>
        <h1>Main heading</h1>
        <p>Some text</p>
    </header>
    <h2>Level 2 heading</h2>
    <p>Some more text</p>
    <h3>Level 3 heading</h3>
    <p>A bit more text</p>
    <h2>Another level 2 heading</h2>
    <p>The last bit of text</p>
</article>
<article>
    <header>
        <h1>Main heading</h1>
        <p>Some text</p>
    </header>
    <h2>Level 2 heading</h2>
    <p>Some more text</p>
    <h3>Level 3 heading</h3>
    <p>A bit more text</p>
    <h2>Another level 2 heading</h2>
    <p>The last bit of text</p>
</article>
```

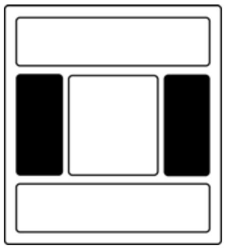



THE <header> ELEMENT DOES NOT HAVE ANY IMPACT ON THE DOCUMENT OUTLINE. IT'S AS IF IT'S NOT THERE.

## Common page elements

There are more new elements than <article>, <section>, <header>, and <hgroup>. Let's look at some more pages from our set of typical websites.


WEB PAGES ARE MORE THAN JUST ARTICLES, SECTIONS, AND HEADINGS. WHAT ABOUT OTHER ELEMENTS?
WE HAVE <nav> FOR NAVIGATION AND <aside> FOR NONESSENTIAL CONTENT LIKE SIDEBARS.
<aside>? THAT SOUNDS LIKE A STAGE DIRECTION FOR A POST MODERN SITCOM. WHY NOT JUST CALL IT SIDEBAR?
BECAUSE THE ELEMENT IS MORE GENERAL PURPOSE. ADS, NAVIGATION GROUPS, OR PULLQUOTES COULD ALSO BE ASIDES.
HMM, IF YOU SAY SO. WHAT ABOUT FOOTERS? THERE MUST BE A <footer> ELEMENT TO GO WITH <header>.
THERE IS, ALONG WITH A <small> ELEMENT FOR FINE PRINT—LEGAL INFORMATION AND DISCLAIMERS.
HANG ON, THERE'S A <nav> ELEMENT IN THAT FOOTER!
YES, THERE'S NO RULE THAT SAYS YOU CAN HAVE ONLY ONE PER PAGE. ANYWHERE YOU HAVE NAVIGATION, YOU CAN USE THE <nav> ELEMENT. LINKS IN THE FOOTER ARE VERY COMMON.
NAV
ASIDE
FOOTER
NAV
SMALL

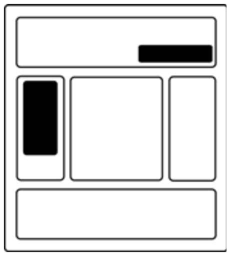


The <aside> element is intended for content that isn't part of the flow of the text in which it appears but is still related in some way. In many books, including this one, you'll see sidebars for things such as terminology definitions and historical background, like the one that follows—these would be marked up as <aside> if the book was HTML5. Sidebars are also common in website design, although the meaning is slightly different: often they contain navigation or related links.

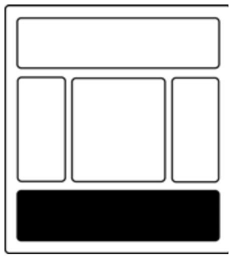
**Sidebar**

This is an example sidebar. If this were an HTML5 document, it would be marked up with the <aside> element.

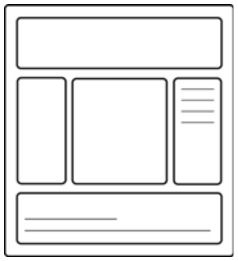
The <nav> element is intended for navigation, both within the page itself, as in the Wikipedia table of contents, and through the rest of the website. You can have any number of <nav> elements on a page. On large sites, it's common to have global navigation across the top (in the <header>) and local navigation in a sidebar (in an <aside> element).



The <footer> element generally appears at the end of a document, a section, or an article. As with the <header> element, its content is generally metainformation—author details, legal information, or links to related information. But it's valid to include <section> elements within a footer—for example, when marking up appendixes.



The <small> element often appears within a <footer> or <aside> element—it contains copyright information, legal disclaimers, and other fine print. Note that it's *not* intended to make text smaller. You may choose to style its contents smaller than your main text, but, as with other elements, its role is to describe its contents, not prescribe presentation.



## The HTML DOCTYPE

The DOCTYPE declaration optionally appears at the start of an HTML document. It comes from the Standard Generalized Markup Language (SGML) that was used to define previous versions of HTML in terms of the language syntax. The DOCTYPE serves two practical functions:

- It's used by HTML validation services to determine which version the document uses.
- Browsers use the DOCTYPE to determine which rendering mode to use.

The rendering modes are Standards, Almost Standards, and Quirks mode. These modes represent various stages in the history of browser development and allow modern browsers to display old web pages the way they were intended. See appendix C for a discussion of these factors—the short version is, Standards mode is what you want.

HTML5 is defined in terms of its DOM representation after parsing, so it doesn't need a `DOCTYPE` for validation, but we still want legacy browsers to render pages in standards-compliant mode. With this in mind, the authors of the HTML5 spec worked out the minimal amount of markup required to trigger Standards mode in browsers:

```
<!DOCTYPE html>
```

Compare this with similar declarations for HTML4 and XHTML1:

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

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

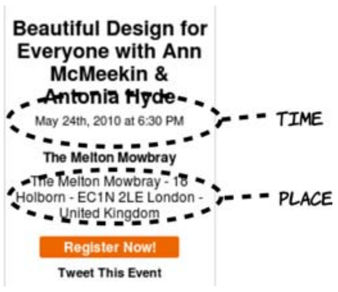
You can see that the HTML5 `DOCTYPE` is much shorter, easier to type, and easier to remember.

## New elements for content

There are several other new or redefined elements in HTML5, and in this section you'll learn about some of them. HTML5 includes dedicated elements for dates as well as figures and captions, all common elements of modern web pages. It also rehabilitates the `<b>` and `<i>` elements that were deprecated in HTML4. This section looks at each of these in turn.

### Time

The `<time>` element allows an unambiguous ISO 8601 date to be attached to a human-readable version of that date. This is useful if you want some other website or service to look at your web pages and extract information. A common use case for this is that you're advertising an event on your website and you'd like it to appear in search results for queries such as "events in London next week." Alternatively, you might decide to write a program to build a timeline of the English monarchy by crawling Wikipedia; being able to parse all the dates in a straightforward way would make this much easier. Following are three examples of these sorts of pages:


Beautiful Design for Everyone with Ann McMeekin & Antonia Hyde

May 24th, 2010 at 6:30 PM

The Melton Mowbray

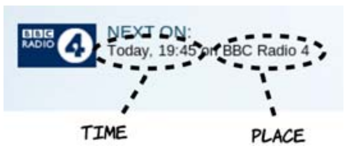
The Melton Mowbray - 18 Holborn - EC1N 2LE London - United Kingdom

Register Now!

Tweet This Event

This WordPress blog is advertising an upcoming event. You can see the key components are all present here:

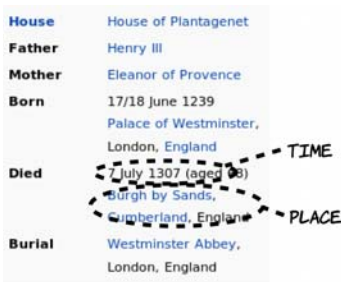
- An event title
- A time
- A place


BBC RADIO 4

NEXT ON:

Today, 19:45 on BBC Radio 4

The BBC website has a page for each program, and this contains information about when the program will next be broadcast. Although the title isn't shown here, you can see the key components: a time and a place (although in this case the "place" is more abstract).



| | |
|---|---|
| House | House of Plantagenet |
| Father | Henry III |
| Mother | Eleanor of Provence |
| Born | 17/18 June 1239 Palace of Westminster, London, England |
| Died | 7 July 1307 (aged 68) Burgh by Sands, Cumberland, England |
| Burial | Westminster Abbey, London, England |

Finally, Wikipedia has events on many pages—in this example, the "event" is the death of Edward I. You may not consider this the same sort of thing as the previous two examples, but it shares the same basic characteristics. This pattern is so common that the microformats movement established a standard way of marking it up called hCalendar.

In the previous examples, it should have been fairly easy for you to pick out the key bits of information even without the big dotted circles, but computers need a more structured form of data. One approach to this is microformats.

## Microformats

*Microformats* are an effort to extend the expressive power of standard HTML by using certain attributes, mostly the `class` attribute, in a standardized way. Popular microformats include hCard, for describing contact information, adr for addresses, and hCalendar for describing events. Similar technologies include the more formal RDFa and HTML5's own microdata (see section 2.5.3).

The main goal of microformats is to render common information like events easily parseable by computers without affecting the end-user presentation.

MICROFORMATS ENABLE A NUMBER OF USEFUL APPLICATIONS: SEARCH ENGINES THAT CAN TELL YOU ABOUT NEARBY UPCOMING EVENTS AND BROWSERS THAT CAN AUTOMATICALLY ADD THE EVENTS TO YOUR CALENDAR.

Addresses, being naturally plain text information with some internal structure (house number, street name, city, and so on) are relatively easy to deal with, as long as there's some way to demarcate the components. Microformats manage this by adding a class of location to the containing element, or alternatively using another microformat, adr, to describe the address in detail. Dates and times are more complicated. Take the simple example 1/6/2011. If you're reading this in the United States, you probably interpret that as January 6, whereas in the UK the date is 1st June. Or have another look at the earlier BBC example: the date is "today." I took that screenshot some time ago—how useful is "today" now? You may think this is no more or less ambiguous than the addresses, but the frustrating thing is that we know that an absolute date and time underlie the more ambiguous human expression that we see more commonly.

COMPUTERS LIKE DATES AND TIMES IN AN UNAMBIGUOUS FORMAT. PEOPLE OFTEN FIND THE UNAMBIGUOUS FORMAT HARD TO DIGEST BUT CAN EASILY UNDERSTAND AMBIGUOUS DATES FROM THE CONTEXT. TO SERVE BOTH, WEB PAGES NEED TO PROVIDE DATES IN TWO FORMATS.

HTML5 solves this problem by providing a `<time>` element. Let's look at an example:

```
<time datetime="2011-06-01">today</time>
```

We can be more specific:

```
<time datetime="2011-06-01T18:00:00+01:00">6 o'clock on 1/6/2011</time>
```

Humans get a readable time that they can disambiguate through the context in the normal way, and computers can read the ISO 6801 date and see the date, time, and time zone.

**Time and data**

Originally the `<time>` element had a `pubdate` attribute to allow for its use in marking up blog posts and other articles. Early in 2012, the entire `<time>` element was removed from the WHATWG version of the spec because it didn't appear to be getting used for that purpose. There was something of an uproar within the community, and the `<time>` element was reinstated shortly after, along with a new element, `<data>`, for more general-purpose association of human-readable text with data for computers. At the time of writing, this new element has not yet made it into the W3C version of the spec, so it isn't covered here.

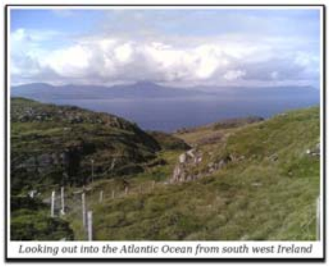
## Images and diagrams with <figure> and <figcaption>

Putting an image in a web page is easy enough: the `<img>` element has existed since the early days of the web. It was somewhat controversial at the time, and several alternatives were put forward; but the most popular browser (Mosaic) implemented it, so it became a de facto standard. The ability to add images was one of the main things that catapulted the web from being an academic document-sharing network into a worldwide phenomenon in the mid 1990s, but since that early take-up not much has changed.

The `<img>` tag is limited from a semantic standpoint—there's no visible way to associate explanatory text with the image. It's possible to use the `alt` and `longdesc` attributes, but because neither is visible by default, both have been somewhat ignored or misused in the real world. The `<figure>` element offers an alternative—it groups the figure with its caption.

This is what the markup for the following screenshot looks like:

```
<figure>
    <img src="scenery.jpg" alt="Picture of the Irish south coast">
    <figcaption>Looking out into the Atlantic Ocean
     from south west Ireland</figcaption>
</figure>
```


Looking out into the Atlantic Ocean from south west Ireland

Note that `<figure>` doesn't have to contain an `<img>` element. It might instead contain an SVG drawing or a `<canvas>` element, or even ASCII art in a `<pre>` element. Whatever type of graphic it contains, the `<figure>` element links the graphic to the caption.

## Emphasizing words and phrases

The `<b>` and `<i>` elements have a long history in HTML. They were listed, along with the `<em>` and `<strong>` elements, in the character-highlighting section of the 1993 IETF draft proposal for HTML. The `<b>` and `<i>` elements are listed in the subsection "Physical Styles" (along with `<tt>`)—that is, their purpose was entirely presentational. Meanwhile, `<em>` and `<strong>` (along with several others) are in the subsection "Logical Styles"—elements with semantic meaning. This early distinction highlights the problem `<b>` and `<i>` would later run into.

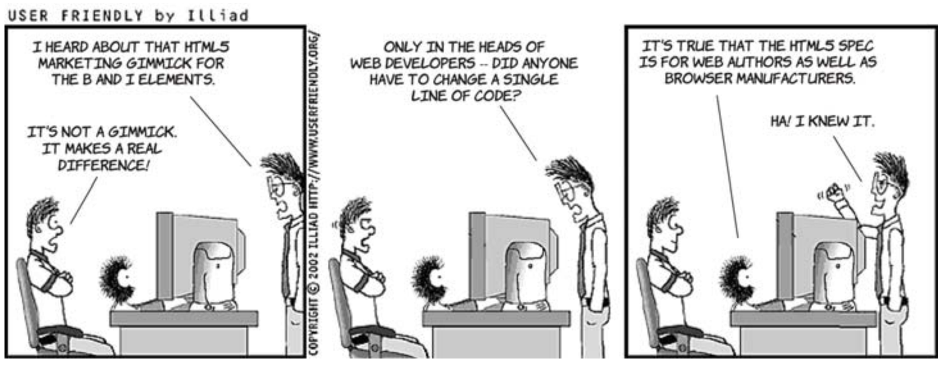
You saw at the start of this chapter that separation of concerns is the Holy Grail of web authoring—HTML for content, CSS for presentation, and JavaScript for behavior. Because `<b>` and `<i>` are entirely presentational, their use has long been frowned on, and there have been several serious proposals to remove them from HTML. Meanwhile, `<strong>` and `<em>` have always had a semantic definition while appearing identical to `<b>` and `<i>`, respectively, in most browsers.

EVER PRAGMATIC, THE HTML5 SPEC RECOGNIZES THAT, WITH MILLIONS OF PAGES OF LEGACY CONTENT OUT THERE, BROWSERS AREN'T GOING TO BE DROPPING SUPPORT FOR `<b>` AND `<i>` ANY TIME SOON. ON THE OTHER HAND, BLINDLY USING `<em>` INSTEAD OF `<i>` AND `<strong>` INSTEAD OF `<b>`, OR USING A `<span>` ELEMENT TO APPLY A BOLD OR ITALIC STYLE TO A WORD ISN'T GOOD PRACTICE SEMANTICALLY.

So, instead of removing either `<b>` or `<i>`, HTML5 redefines and rehabilitates them.

| Element | HTML4 definition | HTML5 definition (taken from the spec on May 12, 2010) |
|---|---|---|
| `<i>` | Renders as italic text style | "The i element represents a span of text in an alternate voice or mood, or otherwise offset from the normal prose, such as a taxonomic designation, a technical term, an idiomatic phrase from another language, a thought, a ship name, or some other prose whose typical typographic presentation is italicized." |
| `<b>` | Renders as bold text style | "The b element represents a span of text to be stylistically offset from the normal prose without conveying any extra importance, such as key words in a document abstract, product names in a review, or other spans of text whose typical typographic presentation is boldened." |

As you can see, the HTML4 definition is entirely presentational, whereas the HTML5 definition goes to great lengths to give a semantic meaning while remaining compatible with the purely presentational uses of the two elements for backward compatibility.


USER FRIENDLY by Illiad
I HEARD ABOUT THAT HTML5 MARKETING GIMMICK FOR THE B AND I ELEMENTS.
IT'S NOT A GIMMICK. IT MAKES A REAL DIFFERENCE!
COPYRIGHT © 2002 ILLIAD HTTP://WWW.USERFRIENDLY.ORG/
ONLY IN THE HEADS OF WEB DEVELOPERS -- DID ANYONE HAVE TO CHANGE A SINGLE LINE OF CODE?
IT'S TRUE THAT THE HTML5 SPEC IS FOR WEB AUTHORS AS WELL AS BROWSER MANUFACTURERS.
HA! I KNEW IT.

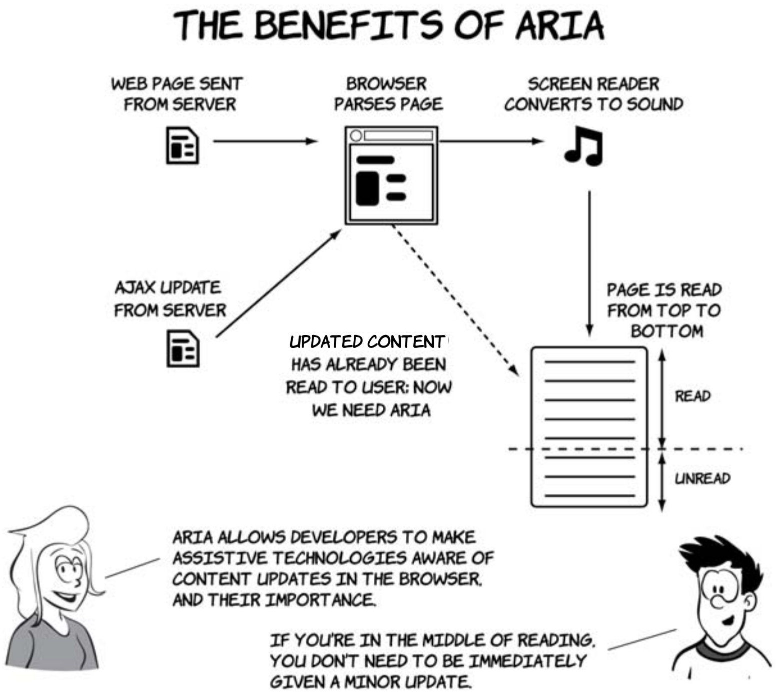
## HTML5's new global attributes

An attribute is *global* if it can be applied to all elements. The two most obvious global attributes in HTML4 are `id` and `class`, which, as you saw in the section "Why do we need new elements?" can be used to add extra semantic information to elements. In this section, you'll learn about new HTML5 global attributes from three major categories:

- Accessibility for Rich Internet Applications (ARIA), for providing extra data to accessibility tools
- `Data-*` attributes, for providing extra data for scripts on your page
- Microdata attributes, for providing extra data to browsers and scripts on other sites

## Accessibility with ARIA

ARIA is a standard developed at the W3C in response to the generally poor accessibility of early AJAX-based web applications.


THE BENEFITS OF ARIA
WEB PAGE SENT FROM SERVER
BROWSER PARSES PAGE
SCREEN READER CONVERTS TO SOUND
AJAX UPDATE FROM SERVER
UPDATED CONTENT HAS ALREADY BEEN READ TO USER: NOW WE NEED ARIA
PAGE IS READ FROM TOP TO BOTTOM
READ
UNREAD
ARIA ALLOWS DEVELOPERS TO MAKE ASSISTIVE TECHNOLOGIES AWARE OF CONTENT UPDATES IN THE BROWSER. AND THEIR IMPORTANCE.
IF YOU'RE IN THE MIDDLE OF READING, YOU DON'T NEED TO BE IMMEDIATELY GIVEN A MINOR UPDATE.

Notifying users of AJAX updates isn't the only benefit ARIA can provide. ARIA consists of a set of attributes and values that can describe to assistive technology the roles of various page elements and their status. In other words, they add semantic value to HTML elements so you can say "this element is a header," "this element is navigation," "this element is a toolbar," and so on. Let's look at an example:

```
<body role="document">
  <div role="note" aria-live="polite"
   aria-relevant="additions removals">
    An update added by JavaScript
  </div>
  <div role="banner">
    <h1 role="heading" aria-level="1">The heading</h1>
  </div>
  <div role="navigation">
    <a role="link" href="/home">Home Page</a>
    <a role="link" href="/inbox">Inbox</a>
  </div>
  <div role="main">
    A very interesting article goes here.
  </div>
  <div role="footer">
    All rights reserved.
  </div>
</body>
```

This should all sound a little familiar to you. What HTML5 aims to accomplish through additions such as the `<header>` and `<nav>` elements is similar to what ARIA tries to accomplish in providing better semantics to assistive technology. But it's still worth bothering with ARIA because it has a wider and more far-reaching vocabulary than HTML5 for describing the components of web applications. Plus it already has wide support among vendors of browsers, operating systems, and assistive technology.

The HTML5 spec has a long list of elements to which user agents should automatically assign particular ARIA roles. These elements are said to have *strong native semantics*, so if you use HTML5 correctly you'll get a certain amount of accessibility for free compared to what HTML4 offered once the browsers and assistive technologies implement support. The HTML5 spec also explicitly lists the allowed ARIA roles for those elements where there's a risk the ARIA role will be in conflict with the HTML5 semantics—these are *implied native semantics*. Validation tools can then flag inappropriate combinations.

Using HTML5, you can cut down on the amount of markup required to provide an accessible user experience. This listing updates the previous one but takes advantage of the strong and implied native semantics in place of several of the ARIA attributes:

```
<body>
  <aside aria-live="polite" aria-relevant="additions removals">
    An update added by JavaScript
  </aside>
  <header role="banner">
    <h1>The heading</h1>
  </header>
  <nav>
    <a href="/home">Home Page</a>
    <a href="/inbox">Inbox</a>
  </nav>
  <article role="main">
    A very interesting article goes here.
  </article>
  <footer>
    All rights reserved.
  </footer>
</body>
```

ALTHOUGH YOU DON'T HAVE TO USE THE IMPLIED ARIA ROLES ON ELEMENTS WITH STRONG SEMANTICS, SUCH AS `<link>` AND `<nav>`, AT PRESENT NO ASSISTIVE TECHNOLOGIES RECOGNIZE THE HTML5 ELEMENTS. YOU SHOULD SPECIFY BOTH FOR BACKWARD COMPATIBILITY.

## Extending HTML with custom attributes

Custom data attributes allow authors to add arbitrary data to elements for their own private use. The idea is that some data isn't directly relevant to the user but does have meaning to the JavaScript on the page that can't be expressed in HTML semantics. It's a standardization of an approach taken by several JavaScript widget libraries, such as Dijit (the Dojo toolkit). These libraries, like HTML5, set out to enhance and extend the application abilities of HTML4—adding things such as combo boxes and date pickers, which HTML5 also provides, but also more complex UI elements such as tree views, drop-down menus, and tabbed containers. Using one of these libraries, you declare an element to be a tab control like this:

```
<div dojoType="dijit.layout.TabContainer">
    <div dojoType="dijit.layout.ContentPane" title="My first tab">
        Lorem ipsum and all around...
    </div>
    <div dojoType="dijit.layout.ContentPane" title="My second tab">
        Lorem ipsum and all around - second...
    </div>
    <div dojoType="dijit.layout.ContentPane" title="My last tab">
        Lorem ipsum and all around - last...
    </div>
</div>
```

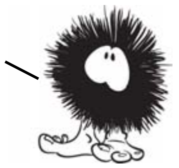
A browser, as with HTML elements, will parse the attribute, even though it doesn't recognize it, and add it to the DOM. The Dijit library will run when the page has loaded, search for these attributes, and run the appropriate JavaScript to enable the advanced control.

It may seem as though everyone has been getting along fine with creating their own attributes, so why add support for custom attributes to HTML5? Well, for one thing, creating your own will stop your markup from validating.

Failing validation may not bother you too much, but if you're looking for that one unintended mistake, having to sift through many intended ones should be unnecessary. Plus there's a risk that the attribute names chosen by the widget libraries will be used in future versions of HTML—after all, one of the goals of the spec is to codify existing common uses.

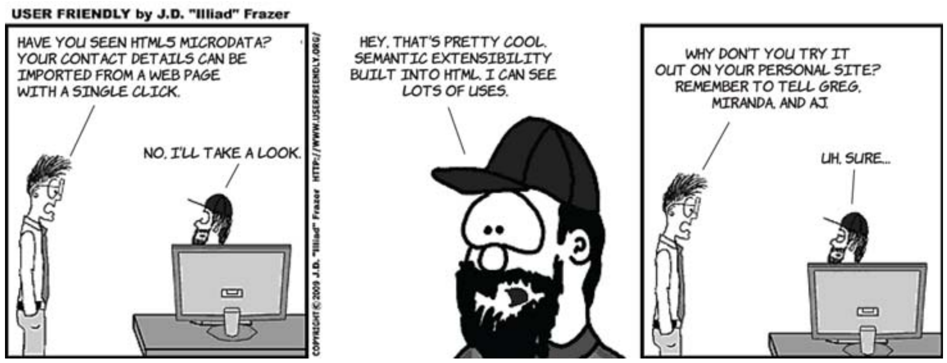
The HTML5 solution to both the validation and potential name-clash issues is the `data-*` attribute collection. The `*` is a wildcard—that is, it can be whatever you want it to be. But anything starting with `data-` will be allowed through the validator, and you're guaranteed that no `data-*` attributes will be made part of HTML.

THE data-* ATTRIBUTES ALLOW YOU TO ADD INFORMATION TO YOUR PAGE FOR YOUR OWN PERSONAL USE. IF YOUR GOAL IS TO SHARE INFORMATION WITH OTHER WEBSITES, YOU SHOULD INSTEAD USE MICRODATA.



## Expressing more than just document semantics with microdata

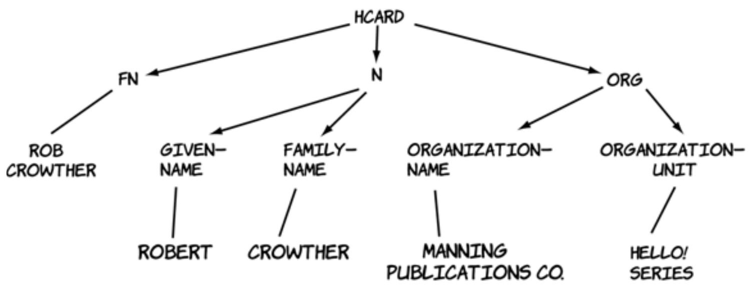
*Microdata* extends the expressive power of HTML to cover things that aren't strictly markup. You can use microdata to designate a portion of your page as describing contact information, a calendar event, or licensing information.


USER FRIENDLY by J.D. "Illiad" Frazer
HAVE YOU SEEN HTML5 MICRODATA? YOUR CONTACT DETAILS CAN BE IMPORTED FROM A WEB PAGE WITH A SINGLE CLICK.
NO, I'LL TAKE A LOOK.
COPYRIGHT © 2009 J.D. "Illiad" Frazer HTTP://WWW.USERFRIENDLY.ORG/
HEY, THAT'S PRETTY COOL. SEMANTIC EXTENSIBILITY BUILT INTO HTML. I CAN SEE LOTS OF USES.
WHY DON'T YOU TRY IT OUT ON YOUR PERSONAL SITE? REMEMBER TO TELL GREG, MIRANDA AND AJ.
UH, SURE...

Microdata uses three global attributes: `item`, `itemtype`, and `itemprop`. All three can be seen in action in this short example that describes contact information:

```
<section id="rob" itemscope
 itemtype="http://microformats.org/profile/hcard">
  <h1 itemprop="fn">Rob Crowther</h1>
  <p itemprop="n" itemscope>Full name:
    <span itemprop="given-name">Robert</span>
    <span itemprop="additional-name">John</span>
    <span itemprop="family-name">Crowther</span>
  </p>
  <p itemprop="org" itemscope>
    <span itemprop="organization-name">Manning Publications Co.</span>
    (<span itemprop="organization-unit">Hello! Series</span>)
  </p>
</section>
```

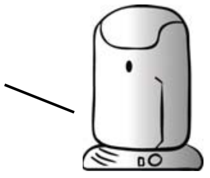
This code, because of the `itemtype` attribute on the parent element referencing the hCard vocabulary, describes a person—me! The `itemprop` attributes are extracted as a set of name-value pairs into a tree-like data structure following the markup, like this:



This information could then be recovered from the page in a usable format by a web browser or a search engine. Of course, you may not want the information to be more easily usable by computers; normal rules of internet publishing apply.

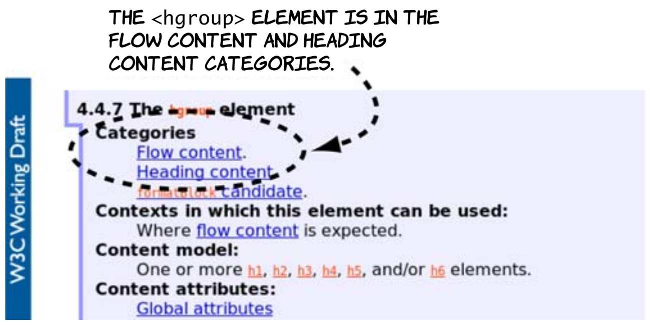
USER FRIENDLY by Illiad
THIS MICRODATA STUFF IS FUN. IT'S REFRESHING FOR STEF TO TAKE AN INTEREST IN TECH WITHOUT MONEY BEING INVOLVED.
http://www.userfriendly.org/
Copyright (c) 2000 Illiad
BUT WHAT CAN HE GAIN FROM GETTING A LOAD OF TECHIES TO PUBLISH THEIR CONTACT INFO IN EASILY PARSEABLE FORM.
UH-OH.
LET'S SEE:
STEP 1 TRICK TECH SAVVY PEOPLE INTO PUBLISHING CONTACT INFO.
2. HARVEST EMAIL ADDRESSES.
3. SELL TO HIGHEST BIDDER.

YOU'LL LEARN MORE ABOUT MICRODATA IN CHAPTER 5 WHEN WE LOOK AT THE MICRODATA API, A CONVENIENT METHOD FOR EXTRACTING THE DATA FROM A DOCUMENT. THE NEXT SECTION LOOKS AT HOW YOU CAN PRODUCE A VALID HTML5 DOCUMENT BY LEARNING ABOUT THE CONTENT MODEL.



## The HTML5 content model

The content model is somewhat theoretical, but it's important because it's the main way of determining whether it's valid to use a certain element at a particular place in your document. In HTML5, elements are split into categories. One element can be a member of several categories; it can also be a member of a category only in particular circumstances, such as when an attribute is given a certain value. In this section, you'll learn where you can find this information in the spec, what elements fit into which content categories, and what the content categories are. The categories of which an element is a member are stated prominently in the HTML5 spec. The following diagram shows the content categories of the `<hgroup>` element.


THE <hgroup> ELEMENT IS IN THE FLOW CONTENT AND HEADING CONTENT CATEGORIES.
W3C Working Draft

**4.4.7 The hgroup element**

**Categories**

Flow content.

Heading content.

formatblock candidate.

**Contexts in which this element can be used:**

Where flow content is expected.

**Content model:**

One or more h1, h2, h3, h4, h5, and/or h6 elements.

**Content attributes:**

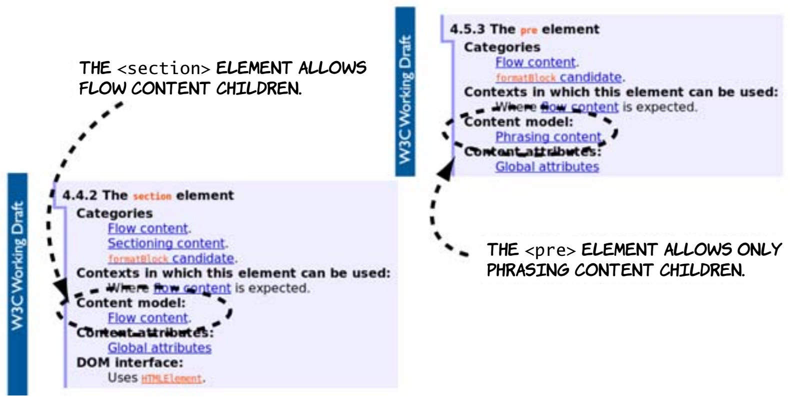
Global attributes

The spec is good if you have a question about a particular element, but it's cumbersome if you want a quick overview. Rather than trawl through the entire spec, the content categories can be summarized in a table.

| Element | Metadata content | Flow content | Phrasing content | Interactive content | Embedded content | Heading content | Sectioning content |
|---|---|---|---|---|---|---|---|
| `<a>`, `<button>`, `<input>`, `<keygen>`, `<label>`, `<select>`, `<textarea>` | | ● | ● | ● | | | |
| `<abbr>`, `<area>`, `<b>`, `<bdo>`, `<br>`, `<cite>`, `<code>`, `<datalist>`, `<del>`, `<dfn>`, `<em>`, `<i>`, `<ins>`, `<kbd>`, `<map>`, `<mark>`, `<meter>`, `<output>`, `<progress>`, `<q>`, `<ruby>`, `<samp>`, `<small>`, `<span>`, `<strong>`, `<sub>`, `<sup>`, `<time>`, `<var>`, `<wbr>` | | ● | ● | | | | |
| `<address>`, `<blockquote>`, `<div>`, `<dl>`, `<fieldset>`, `<figure>`, `<footer>`, `<form>`, `<header>`, `<hr>`, `<ol>`, `<p>`, `<pre>`, `<table>`, `<ul>`, `<Text>` | | ● | | | | | |
| `<article>`, `<aside>`, `<nav>`, `<section>` | | ● | | | | | ● |

| Element | Metadata content | Flow content | Phrasing content | Interactive content | Embedded content | Heading content | Sectioning content |
|---|---|---|---|---|---|---|---|
| `<audio>`, `<embed>`, `<iframe>`, `<img*>`, `<object>`, `<video>` |  | ● | ● | ● | ● |  |  |
| `<base>`, `<title>` | ● |  |  |  |  |  |  |
| `<canvas>`, `<math>`, `<svg>` |  | ● | ● |  | ● |  |  |
| `<command>`, `<link>`, `<meta>`, `<noscript>`, `<script>` | ● | ● | ● |  |  |  |  |
| `<details>`, `<menu>` |  | ● |  | ● |  |  |  |
| `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>`, `<hgroup>` |  | ● |  |  |  | ● |  |
| `<style>` | ● | ● |  |  |  |  |  |

Now you know which content models apply to which elements, but that's only part of the story. You also need to know what content categories are allowed as children of any given element. The following diagram shows a couple of other excerpts from the HTML5 spec to illustrate where you can find this information.


THE <section> ELEMENT ALLOWS FLOW CONTENT CHILDREN.
THE <pre> ELEMENT ALLOWS ONLY PHRASING CONTENT CHILDREN.
W3C Working Draft
W3C Working Draft

4.5.3 The pre element
Categories
Flow content.
formatBlock candidate.
Contexts in which this element can be used:
Where flow content is expected.
Content model:
Phrasing content.
Content attributes:
Global attributes

4.4.2 The section element
Categories
Flow content.
Sectioning content.
formatBlock candidate.
Contexts in which this element can be used:
Where flow content is expected.
Content model:
Flow content.
Content attributes:
Global attributes
DOM interface:
Uses HTMLElement.

The different content types aren't applied arbitrarily; each has a distinct meaning. The following table summarizes the different types.

| | |
|---|---|
| **Flow content** | Most elements are categorized as flow content. It's the default content type for elements visible on the page. |
| **Sectioning content** | Sectioning content defines the scope of headers and footers and feeds into the document outline. |
| **Heading content** | Heading content, as you might expect, is just for headings and `<hgroup>`. |
| **Phrasing content** | Phrasing content is mostly used to describe the text of a document. In most cases, phrasing content can only contain other phrasing content. |
| **Embedded content** | Embedded content is used to put an external resource into the web page—for example, an image or video. |
| **Interactive content** | Interactive content is elements that are specifically intended for user interaction—mostly form controls. Note that other elements can be made responsive to user input through the use of JavaScript, but elements categorized as interactive content have default functionality in the browser. |
| **Metadata content** | Metadata content sets up the presentation or behavior of the rest of the content, or sets up the relationship of the document with other documents, or conveys other out-of-band information. |

Now that you know all about the content model, you'll be able to use the HTML5 spec to write valid HTML5 documents. That's more than enough theory for now. The next section gets back to practicalities and considers whether your users' browsers will support HTML5 and what to do about it if they don't.

## Browser support

Do the new elements we've discussed in this chapter work in today's browsers? The short answer is, yes (with a couple of exceptions); the

FOR A TEXT ELEMENT LIKE <p>, WHICH ISN'T REQUIRED TO DO MUCH EXCEPT APPEAR ON THE PAGE, THERE ARE TWO PRINCIPAL REQUIREMENTS:

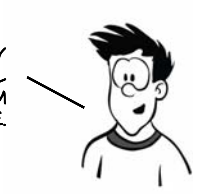
- IT SHOWS UP IN THE DOM WITH AT LEAST A STANDARD SET OF ELEMENT PROPERTIES.
- IT SHOWS UP IN THE USER'S BROWSER WITH SOME SORT OF DEFAULT PRESENTATION.

long answer is a little more complex. Consider this question: what does it mean to say that a browser supports the `<p>` element?

It turns out that the first requirement is easy to satisfy—as long as you follow simple tag-naming rules, you can put any tags in your HTML, and all browsers will put the tags in the DOM with a default set of properties.

Where problems arise is with regard to the second requirement: having a default presentation. Browsers have only recently started providing any default presentation for the new elements in HTML5; for instance, Firefox 3.6 doesn't, but Firefox 4.0 does. But this isn't much of a problem. As you know, we web authors define our content in HTML and our presentation in CSS—and browsers work exactly the same way. The default presentation for the supported elements is defined in CSS. If you use Firefox, you can even find this file on your hard drive—it's called html.css.

USING THESE NEW ELEMENTS IS A MATTER OF TAKING ON THE RESPONSIBILITY OF PROVIDING SOME DEFAULT CSS RULES FOR THEM. IN MOST CASES YOU'LL WANT TO WRITE CSS FOR THESE ELEMENTS ANYWAY, SO THIS DOESN'T SEEM LIKE TOO MUCH EFFORT. LET'S SEE HOW IT WORKS WITH AN EXAMPLE.

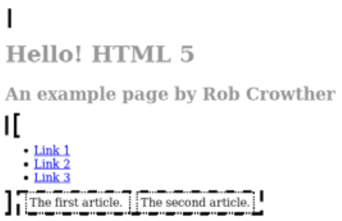


Here's a simple HTML5 document to experiment with:

```
<header>
  <hgroup>
    <h1>Hello! HTML 5</h1>
    <h2>An example page by Rob Crowther</h2>
  </hgroup>
</header>
<nav>
  <ul>
    <li><a href="#">Link 1</a></li>
    <li><a href="#">Link 2</a></li>
    <li><a href="#">Link 3</a></li>
  </ul>
</nav>
<section>
  <article>The first article.</article>
  <article>The second article.</article>
</section>
```

Starting with the following basic styles, this screenshot shows what the page looks like in a browser that doesn't have any default HTML5 styles:

```
header, nav, section, article
 {padding: 4px; margin: 4px;}
header
 { background: #000; color: #999; }
nav
 { border: 4px solid #000; }
section
 { border: 4px dashed #000; }
article
 { border: 2px dotted #000; }
```



Hello! HTML 5

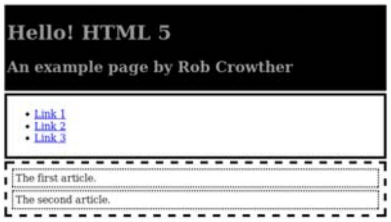
An example page by Rob Crowther

- Link 1
- Link 2
- Link 3

The first article. The second article.

By making a single change to that CSS, you can make the page work in most older browsers. See if you can spot it:

```
header, nav, section, article
 { padding: 4px; margin: 4px;
   display: block; }
header
 { background: #000; color: #999; }
nav
 { border: 4px solid #000; }
section
 { border: 4px dashed #000; }
article
 { border: 2px dotted #000; }
```



Hello! HTML 5

An example page by Rob Crowther

- Link 1
- Link 2
- Link 3

The first article.

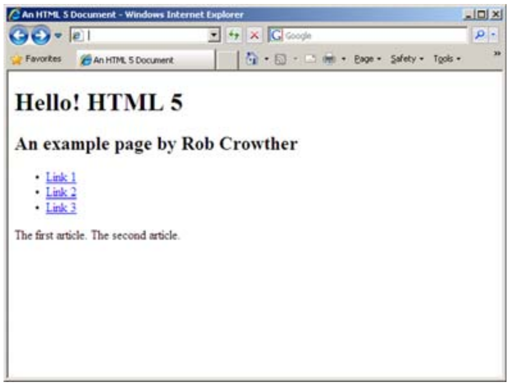
The second article.

If you specify that the block-level HTML5 elements `<header>`, `<nav>`, `<section>`, and `<article>` should be `display: block`, everything works as you want.

Most of the major browsers work identically in this regard. Unfortunately, there are two exceptions, one minor and one major. The minor one is Firefox 2.0; Firefox users tend to upgrade regularly, so this version is now used by a very small number of people and we won't worry about it. The larger problem is Internet Explorer 8 and earlier, which is still one of the most commonly used browsers on the web.

## Supporting Internet Explorer

Internet Explorer won't apply CSS rules to any elements it doesn't recognize. Here's what the sample page looks like in IE7.



Hello! HTML 5

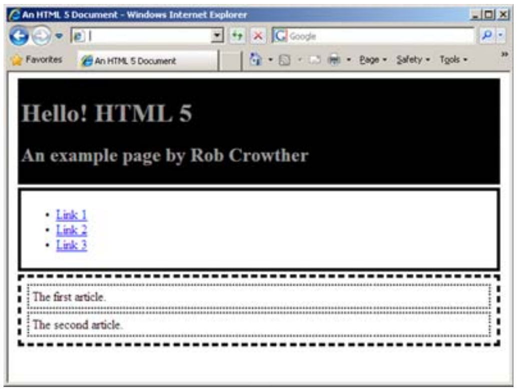
An example page by Rob Crowther

- Link 1
- Link 2
- Link 3

The first article. The second article.

But all is not lost. You can trick IE into recognizing elements with a bit of JavaScript. This code will persuade IE that the `<section>` element exists and should have styles applied to it:

```
document.createElement("section");
```



Hello! HTML 5

An example page by Rob Crowther

- Link 1
- Link 2
- Link 3

The first article.

The second article.

Here's the final listing, with each element we want to use enabled in IE:

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

<style>
  header, nav, section, article {
   padding: 4px; margin: 4px; display: block; }
  header { background: #000; color: #999; }
  nav { border: 4px solid #000; }
  section { border: 4px dashed #000; }
  article { border: 2px dotted #000; }
</style>
```

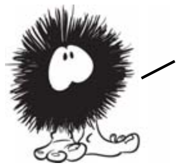
### Enabling HTML5 support in Internet Explorer with html5.js

Rather than work out for yourself what elements you need to fix in Internet Explorer, you can use one of the freely available compatibility scripts. A simple one with a good following is html5.js, available at http://code.google.com/p/html5shiv/.

Of course, the main drawback of these approaches is that they won't work if JavaScript is disabled in the browser or if something blocks your JavaScript from being downloaded, such as a corporate content filter or a personal firewall. Although this is likely to be a small percentage of users for most sites, you should do some analysis of your existing site visitors before embarking on an HTML5 redesign.

## Summary

In this chapter, you've learned about the new markup elements in HTML5 and the formal structure provided for them, and the elements inherited from HTML4, provided by the outlining algorithm and the content model. You've looked at several popular websites and seen how the content they display fits naturally into the new semantic elements of HTML5, reducing the need for content authors to add semantic meaning to neutral `<div>` and `<span>` tags through the `id` and `class` attributes. You've also seen how the new global attributes in HTML5 allow you to extend the expressive power and accessibility of HTML documents.



NOW THAT YOU'VE LEARNED HOW HTML5 IMPROVES MATTERS FOR THOSE WRITING TRADITIONAL HTML DOCUMENTS, IT'S TIME TO MOVE ON TO THE MAIN FOCUS OF HTML5: MARKUP FOR APPLICATIONS. WE'LL START IN THE NEXT CHAPTER WITH A LOOK AT THE ENHANCED SUPPORT FOR FORMS.

WEB DEVELOPMENT/HTML

Free eBook
see insert

# HTML5 & CSS3

Rob Crowther

"A fast-paced introduction. Recommended to anyone who needs a quick-start resource."
—Jason Kaczor, Microsoft MVP

"It's 2012. You need this book!"
—Greg Donald, CallProof, LLC

"Everything you need to know explained simply and clearly."
—Mike Greenhalgh, NHS Wales

"Level up your web skills!"
—Greg Vaughn, LivingSocial

Whether you're building web pages, mobile apps, or desktop apps, you need to learn HTML5 and CSS3. So why wait? **Hello! HTML5 & CSS3** is a smart, snappy, and fun way to get started now.

In this example-rich guide to HTML5 and CSS3, you'll start with a user-friendly introduction to HTML5 markup and then take a quick tour through forms, graphics, drag-and-drop, multimedia, and more. Next, you'll explore CSS3, including new features like drop shadows, borders, colors, gradients, and backgrounds. Every step of the way, you'll find hands-on examples, both large and small, to help you learn by doing.

PROFESSIONAL DEVELOPMENT? AREN'T MOST WEBSITES DONE BY THE BOSS'S TEENAGE NEPHEW?
YOU KNOW IT'S NOT 1999 ANY MORE. RIGHT?

**What's inside**

- Easy-to-follow intro to HTML5 and CSS3
- Fully illustrated and loaded with examples
- Designed for low-stress learning
- No prior experience needed!

Don't worry—you aren't alone! The cast of characters from ***User Friendly*** is learning HTML5 and CSS3 along with you as you read.

**Rob Crowther** is a web developer and blogger from London.

To download their free eBook in PDF, ePub, and Kindle formats, owners of this book should visit manning.com/HelloHTML5andCSS3

ISBN 13: 978-1-935182-89-4
ISBN 10: 1-935182-89-7
53999
9 781935 182894

MANNING US $39.99 / Can $41.99