

## CHAPTER 4



# Structural HTML Elements

I briefly introduced the basic HTML syntax in Chapter 1 and then explained the information inside the head element. There is no displayable content in the head element; the data there is intended for applications to use such as browsers, screen readers, or search engines. Now we turn our attention to the body element; this is where the content goes.

## Content Categories

There are over 100 elements defined in HTML5. I've explained a few of them in Chapter 1 and over the next 5 chapters I'll demonstrate the remaining ones. Elements are organized into content categories and there are general rules, based on these categories, which define where an element can be used and what it can contain. An element can be in more than one category and there are a few elements that don't belong in any category.

- *Metadata* - These were explained in Chapter 1; they have no real content but provide metadata about the HTML document and information to applications that will process the document.
- *Sectioning* - These elements are used to organize a page into sections, and are also used to construct a document outline.
- *Heading* - These elements are used within sections to define the title and subtitles of a section.
- *Embedded* - Embedded elements are used to insert non-HTML content, such as images, within the document. These will be described in Chapter 7.
- *Interactive* - These elements provide for user interaction. A button or an input field are common examples of this.
- *Form* - Form elements are used for capturing user input and are further divided into several subcategories. I will explain these in Chapter 8.
- *Phrasing* - These elements are used to mark up text, or rather phrases, which are combined to form paragraphs. These elements will be demonstrated in Chapter 5.
- *Flow* - The vast majority of elements will fall into this category. These are elements that have actual content such as text or some embedded content like images or video. The term “flow” is used because these elements take up space and generally flow from one element to the next, across, or down the page.

The reference material in Appendix C includes an alphabetical list of all of the HTML5 elements that indicates which categories each is in.

---

■ **Note** Most of the elements that are not in any content category are used only as child elements for a specific element. For example, the `table` element is in the flow category, but the child elements such as `tr` and `td` are not assigned a category as they are only used within a `table`.

---

This chapter will explain the use of sectioning and header elements to organize your HTML document.

## Sectioning Content

It's a good idea to organize your HTML document into logical sections, especially for larger documents. Prior to HTML5 the division (`div`) element was used to group content and they can be nested like this:

```
<div>
  <div>
    <div>
      </div>
    </div>
  </div>
</div>
```

Unfortunately, a `div` can be used to group content for a lot of different reasons, which are not obvious to the reader. HTML5 introduces several new elements that provide more specific semantic groupings. Each of these elements are used to group content into a larger unit. Each one, however, groups the content for a different reason. You could accomplish the grouping with the generic `div` element; the reason to use the more specific element is to make it clear *why* the group is being made. So pay close attention to the specific purpose of each element and use the correct one depending on what you're trying to accomplish.

## Section

The `section` element is used to organize content into logical sections. Think back to an essay you may have written for a school assignment where you'd have an introduction, three main points, and a conclusion. Each of these would be represented in HTML as a `section`. Also, `section` elements can be nested just like `div` elements. If each of your main points have subpoints, you can use a `section` for each subpoint.

There are two guiding principles in choosing your `section` elements. First, sections should be topical; you're organizing content based on the material being presented. Second, sections should flow linearly. For example, the introduction flows into the first main point, and so on. The antithesis of this will be more apparent when I discuss the `aside` element.

## Article

An `article` element is used to group content that can stand on its own. This is typically used when the content is being reused. If you pick up your local Sunday newspaper and read your favorite comic strip, you may not be aware of this, but the same comic strip is included in newspapers all over the world. The same concept is true of web sites, where syndicated content is included in multiple pages. These should be placed in an `article` element because the content is independent of the rest of the page.

However, one of the most common uses of the `article` element is on blogs. Each post on a blog is generally stand-alone content and is often grouped into an `article` element. Also, comments that are posted to a blog are also typically placed in an `article` element.

You can use an `article` element anywhere you would use a section element, if the content is independent and reusable. An article, especially a larger one, will often use the section element to organize the article topically as I described previously. Also, an `article` element can include other `article` elements, as would be the case with a blog post including comments.

## Aside

An `aside` element is used to group content that does not belong in the normal flow of your document. This could be supporting information provided for reference purposes, or perhaps information about the author. It could also be unrelated information such as advertising space or a calendar of events.

The `aside` element is often presented as a sidebar so it doesn't interrupt the flow of the other content. However, the CSS will determine that; the job of a content author is to indicate that a group of content is not part of the normal content flow. And this is done with the `aside` element.

## Nav

The `nav` element is used to group a set of links. A typical example is where you have some sort of menu on the page to jump to internal bookmarks or other related pages. Another example is where you provide links for more information or related material.

You don't have to put every link into a `nav` element. But if you do have content that is comprised primarily of links, put this in a `nav` element. This will indicate that this section of content provides navigation.

## Address

The `address` element is not included in the sectioning content. I am describing it here because it is used to provide contact information for either the entire document or for a specific article. To use it for a single article it should be placed somewhere inside the `article` element. If it applied to the document, it must be inside the `body` element; this is often placed in the `footer` element.

---

■ **Tip** The `address` element is intended for providing contact information related to a document or article. This can include an email address, URL, phone number, mailing address, or any other method of communicating with the author. It may have been better named **contact** because that is what it should be used for. You should not use it to describe an address unless the address is provided as contact information.

---

Here is an example of including an `address` element in a footer:

```
<footer>
  <p>Closing content</p>
  <address>
    <p>Provided by
      <a href="mailto:mcollins@theCreativePeople.com">Mark J. Collins</a>
    </p>
    <p>For more information
      <a href="www.theCreativePeople.com">visit his website</a>
    </p>
  </address>
</footer>
```

The default style for an address element is to display the text in italics. The default styling of this footer will look like Figure 4-1.

**Closing content**

*Provided by [Mark J. Collins](#)*

*For more information [visit his website](#)*

**Figure 4-1.** The default address style

## Outlines

There is a notion in the HTML5 specifications regarding document outlines that is referred to as the outline algorithm. The body element, which is the root element for your content, creates a topmost node in the document outline. Adding any of the sectioning elements, section, article, aside, or nav, will create a new section in the outline. Embedding additional sectioning elements will add more nodes to the outline. The idea here is that by simply nesting sectioning elements, an explicit outline is created for your document.

## Explicit Sections

Let's test this out by creating an HTML document with some nested sections. Creating an outline without any labels is not very interesting. So we'll add an h1 element within each section to give it a name. A sample document is shown in Listing 4-1.

**Listing 4-1.** Creating a document outline using sections

```
<body>
  <h1>My Sample Page</h1>
  <nav>
    <h1>Navigation</h1>
  </nav>
  <section>
    <h1>Top-level</h1>
    <section>
      <h1>Main content</h1>
      <section>
        <h1>Featured content</h1>
      </section>
      <section>
        <h1>Articles</h1>
        <article>
          <h1>Article 1</h1>
        </article>
      </section>
    </section>
  </section>
  <aside>
    <h1>Related content</h1>
```

```

    <section>
      <h1>HTML Reference</h1>
    </section>
    <section>
      <h1>Book list</h1>
      <article>
        <h1>Book 1</h1>
      </article>
    </section>
  </aside>
</section>
</body>

```

If you render this in a browser using only the default styles, the page would look like Figure 4-2.

# My Sample Page

## Navigation

### Top-level

#### Main content

#### Featured content

#### Articles

#### Article 1

#### Related content

#### HTML Reference

#### Book list

#### Book 1

**Figure 4-2.** Sample document with default styles

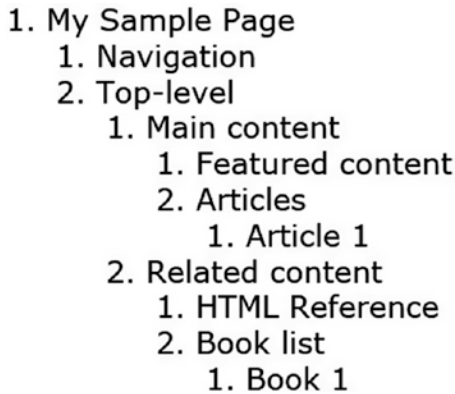
Notice the font for each of the labels is different, even though the same h1 element was used for all of them. The style is automatically updated based on where the element is in the document outline.

---

■ **Note** The `aside` element is below the main section rather than to the side. The alignment of the sections will be controlled through CSS. I will explain this in Chapter 13.

---

To further illustrate this, there is a handy web page that will read your HTML document and display the outline for it. You can find the site at <https://gsnedders.html5.org/outliner>. Paste in your HTML document and click the “Outline this” button. The outline that is displayed will be similar to Figure 4-3.

- 
- ```
graph TD
    1[1. My Sample Page] --> 1_1[1. Navigation]
    1 --> 2[2. Top-level]
    2 --> 2_1[1. Main content]
    2 --> 2_2[2. Related content]
    2_1 --> 2_1_1[1. Featured content]
    2_1 --> 2_1_2[2. Articles]
    2_1_2 --> 2_1_2_1[1. Article 1]
    2_2 --> 2_2_1[1. HTML Reference]
    2_2 --> 2_2_2[2. Book list]
    2_2_2 --> 2_2_2_1[1. Book 1]
```
- 1. My Sample Page
    - 1. Navigation
    - 2. Top-level
      - 1. Main content
        - 1. Featured content
        - 2. Articles
          - 1. Article 1
      - 2. Related content
        - 1. HTML Reference
        - 2. Book list
          - 1. Book 1

**Figure 4-3.** Viewing the document outline

## Document Headings

HTML5 supports document heading elements. In addition to the `h1` element that I used in this example, there are also `h2`, `h3`, `h4`, `h5`, and `h6` elements. Each of these is used to indicate that the heading belongs at the specified level in the document outline. The default styling of the `h1`–`h6` elements is the same that was used with the explicit outline document.

Prior to HTML5, the sectioning elements did not exist. However, using the `h1`–`h6` elements, you could provide implicit sections. If the previous heading used an `h2` element and the current heading uses `h3`, a new section is implicitly defined. Similarly, going from `h3` to `h2` closes the current section.

To create a document with a similar outline, you can create implicit sections using the `h1`–`h6` elements. Instead of using an `h1` element everywhere, you would use a different element to indicate its level in the hierarchy. The following HTML will generate the exact same outline (and output in the browser):

```
<body>
  <h1>My Sample Page</h1>
  <h2>Navigation</h2>
  <h2>Top-level</h2>
  <h3>Main content</h3>
  <h4>Featured content</h4>
  <h4>Articles</h4>
```

```

<h5>Article 1</h5>
<h3>Related content</h3>
<h4>HTML Reference</h4>
<h4>Book list</h4>
<h5>Book 1</h5>
</body>

```

The official W3C recommendation, however, is to not rely on the outlining of the sectioning elements. Figure 4-4 shows the warning displayed in the W3C documentation (<http://www.w3.org/TR/html5/sections.html>).

**⚠Warning!** There are currently no known implementations of the outline algorithm in graphical browsers or assistive technology user agents, although the algorithm is implemented in other software such as conformance checkers. Therefore the [outline](#) algorithm cannot be relied upon to convey document structure to users. Authors are advised to use heading [rank](#) ([h1-h6](#)) to convey document structure.

**Figure 4-4.** Warning about outline algorithm

You should definitely use the sectioning elements to organize your HTML document into sections. This warning is recommending that you also use the corresponding h1–h6 element depending on the document hierarchy where this is feasible.

The `article` element is appropriate for reusable content. In this case, the content is usually not in the same file as the main HTML document. It may be read from a database, provided in some sort of content feed, or extracted from a separate file. It will probably have no knowledge of the document outline in which it will be inserted. The article should start at the h1 level, and include additional sectioning elements as needed. When the article is included in a document, the actual outline level will be adjusted depending on the current level where it is inserted.

---

■ **Note** The specific algorithm for creating an outline is explained in the W3C recommendation, which you can find at this address: <http://www.w3.org/TR/html5/sections.html#outlines>

---

## Header and Footer

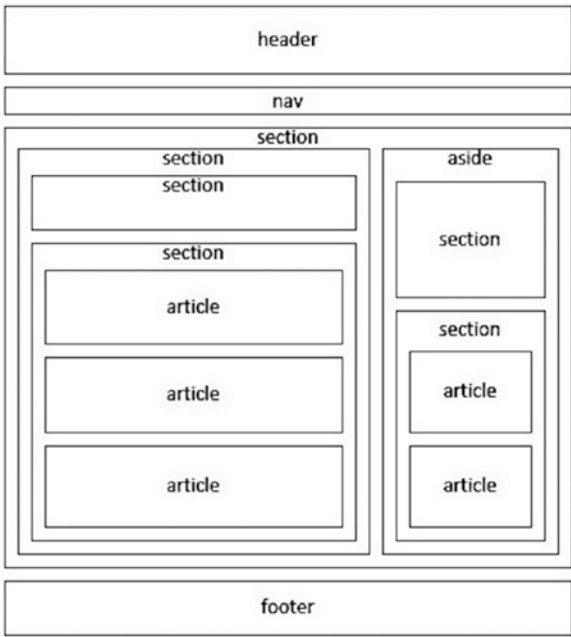
While organizing your page, you should also consider adding a header element at the top and a footer element at the bottom. These elements allow you to group introductory or concluding content for a section of the document.

Unlike the sectioning elements such as `article` or `section`, the header and footer elements do not create a new section in the document outline. Rather they are used to group content for the section that they are placed in. You will typically use a header and a footer element in the `body` element to define the page header and footer. They can also be placed inside a child section such as a `section` element. In this case, they will group introductory content for the section only.

# Planning the Page Layout

Before creating a new web page, it's a good idea to sketch out the basic page structure. This will help you visualize the overall layout and see how the elements are nested together.

The page that I will demonstrate in this chapter will use header and nav elements at the top and a footer element at the bottom. The main area in the middle will use a section element and have two side-by-side areas, each with a series of article tags. The larger area will be enclosed within another section element and provide the primary content, which is organized into article elements. The smaller area, on the right, will use an aside element and will contain a section element. This will contain a series of article elements that will present related information. Figure 4-5 illustrates the page layout.



**Figure 4-5.** Planning the page layout

**Note** This diagram shows spaces between each of the elements to make it easier to understand. In the actual web page, in most cases this space is removed by setting the `margin` attribute to 0.



## Sectioning Roots

There are a few HTML elements that are called sectioning roots, which have their own outline that does not contribute to the outline of the rest of the document. The body element is one of these elements, although this is arguably a special case; the outline of the body element is the document's outline. The other elements in this category include `blockquote`, `details`, `fieldset`, `figure`, and `td`. I'll explain the `fieldset` element in Chapter 8 and the `td` element in Chapter 6.

### Blockquote

The `blockquote` element is used when you need to include a long quotation in a document. The contents of the quote are placed inside the `blockquote` element and can consist of multiple elements including header text, paragraphs, and embedded content. A simple quotation might look like this:

```
<blockquote cite="www.apress.com">
  <h1>Quotation</h1>
  <p>This is a quotation</p>
</blockquote>
```

The header text (h1–h6) will define sections within the quotation, but these will not be part of the document outline. You can verify this yourself by adding a `blockquote` element to your document and running it through the outlining tool I mentioned earlier.

In addition to the global attributes, the `blockquote` element supports the `cite` attribute. Use this to identify the URL of the source of the quote or a resource containing information about the quote.

---

■ **Tip** The `cite` *attribute* provides details about the quote but it is not displayed, at least not normally. The browser could use this information but it's basically provided as metadata. To display the source or a citation you should use the `cite` *element*, which will be explained in the next chapter.

---

### Details

The `details` element allows you to create collapsible sections of content. Inside the `details` element you can include an optional `summary` element that will contain the content that is displayed when the element is collapsed. If no `summary` element is used, the collapsed text will be “Details.”

The remaining content of the `details` element will be hidden when collapsed. The initial state of the `details` element will be collapsed. If you want the contents to be displayed when the page is loaded, add the `open` Boolean attribute.

```
<details open>
  <summary>This is the collapsed text</summary>
  <h1>Details</h1>
  <p>These are collapsable details</p>
</details>
```

Again, like the other sectioning root elements, you can include h1–h6 elements in the `details` element, which will define an outline for the details. However, these sections are not included in the document outline.

---

■ **Caution** As of this writing, the `details` element is not supported by Internet Explorer or Edge and is not fully supported by Firefox.

---

## Figure

The figure element is used to group content that is self-contained and can be logically moved to a different location without affecting the main flow of the document. A unique feature of the figure element is the ability to include a caption within the content.

A figure element is typically used to group an image or some other embedded content along with a caption. It can also be used to group text, such as a code listing along with a caption.

To add a caption, include the `figcaption` element within the figure element. The `figcaption` element must be the first or last child element in the figure element. The caption will be above the content if the `figcaption` element is the first child element. Otherwise, it will be below the content.

Here's a simple example of a figure that includes an embedded image. This is rendered in Chrome as shown in Figure 4-6.

```
<figure>
  <h1>Figure</h1>
  
  <figcaption>Official HTML5 Logo</figcaption>
</figure>
```

Figure



**Figure 4-6.** Using `figure/figcaption` elements

## Grouping Elements

The preceding section described the elements used for organizing an HTML document into smaller sections and those elements related to the document outline. Now I'll explain the remaining grouping elements. These elements are used for primarily semantic purposes and do not affect the outline.

### Paragraph

The paragraph (`p`) element is used to define a paragraph, which is a section of text that contains a single thought or idea. It is normally visually distinct from other paragraphs. For example, in this book the first line of most paragraphs is indented so that paragraphs stand out. The default style of the `p` element will set the margin so there is some extra white space between paragraphs.

The `p` element is probably one of the most overused elements. I can't stress this enough - the element that is chosen should be based on its semantic meaning. A paragraph is a distinct portion of a document that contains a single thought or idea. If that describes the content you are adding, use a `p` element. If not, keep looking. The next chapter will describe the phrasing elements and one of these may be more appropriate.

If you use the appropriate element, the default styling will usually make sense as well. But you're not selecting the element based on the style that it is given. The styles can be easily changed through CSS as long as the appropriate elements are consistently used.

### Horizontal Rule

In HTML4, the `hr` element was used to create a horizontal rule, or line. Since HTML5 provides more semantic definition to its elements, this has been redefined as a thematic break. This is usually placed between paragraphs when there is a change of topic.

The `hr` element is still rendered as a horizontal line for backward compatibility. Of course this can be modified through CSS. There is no content within the `hr` element and the closing tag is not needed. Add an `hr` element like this:

```
<p>paragraph 1</p>
<hr />
<p>paragraph 2</p>
```

### Preformatted

Content placed inside a preformatted (`pre`) element will be rendered just like it is entered, including white space. As I explained in [Chapter 1](#), white space, including carriage returns, tabs, and extra spaces, are normally ignored by the browser. Use the `pre` element to tell the browser to include them. The default style will use a fixed-spaced font, which also aids in preserving the format.

---

■ **Note** The `pre` element is one of the rare exceptions where its use is not primarily for semantic reasons.

---

The `pre` element is used to include content that is already formatted and you want to maintain that format. Including a snippet of code is a good example. Another common use of the `pre` element is when including poetry. For example:

```
<pre>I heard the bells on Christmas Day
Their old, familiar carols play,
    And wild and sweet
    The words repeat
Of peace on earth, good-will to men!</pre>
<cite>Henry Wadsworth Longfellow - 1863, public domain</cite>
```

This is rendered in the browser as shown in Figure 4-7.

```
I heard the bells on Christmas Day
Their old, familiar carols play,
    And wild and sweet
    The words repeat
Of peace on earth, good-will to men!

Henry Wadsworth Longfellow - 1863, public domain
```

**Figure 4-7.** Using the *pre* element

Try replacing the *pre* element with a standard paragraph (*p*) element. The content will be displayed on one line, wrapping as necessary based on the page width.

---

■ **Note** One bit of trivia here, but this poem was written by Henry Wadsworth Longfellow during the American Civil War. After recently losing his wife, he then learned that his son Charles was wounded in battle. It was during the Christmas season and hearing the church bells playing carols he began to wonder if there really was any peace or goodwill. This poem reflects the author's struggle with this, concluding in the end, that as long as the church bells keep playing, there was still hope.

---

## Main

The *main* element is used to indicate that its contents present the primary purpose or topic of the document. This identifies the core, the central theme of the document. As such, the *main* element cannot be inside an *article*, *aside*, *footer*, *header*, or *nav* elements, since these are used for ancillary content. Also you can only have one *main* element per document, which should be fairly obvious. The content in the *main* element should be unique to the document and not shared by other documents.

Of course, the *main* element can be divided into sections as appropriate. Although there is no restriction on including an *article*, *aside*, or *nav* element within a *main* element, this is generally considered poor form. An *article* contains reusable or stand-alone material so it will seldom be unique to the document. An *aside* element is used for content that is not part of the main flow and is likely not part of the central theme that the *main* element is reserved for. However, if the content of *aside* or *article* elements satisfies the criteria of a *main* element, you can use them as well.

There is no default styling for a *main* element; its use is purely semantic. Using it helps applications such as screen readers or search engines go straight to the core content of your document.

## Division

I saved the `div` element for last. As I mentioned earlier, prior to HTML5, the `div` element was used for all types of groupings. Now we have new elements to use for specific purposes: topically (`section`), independent or reusable content (`article`), outside of normal flow (`aside`), and navigation (`nav`). There are also several elements that are available for semantic grouping such as the `main`, `figure`, and `blockquote` elements. The `div` element is used for all other grouping reasons.

One common use of the `div` element is to apply styles to all of its child elements. Most style attributes are inherited from the parent element. So setting an attribute on a `div` element will propagate down to all the elements within the `div`. For example, you may need to hide or show some content based on user input or the state of the web page. By putting all of this content in a single `div`, you can simply update the `div` attributes.

## Listing Elements

Now we'll look elements that are used for listing contents, such as a bulleted list. However, lists in HTML5 are not limited to just a series of bullets or numbered items. Any time you have a list of things, even if the "things" are large blocks of content, you should consider creating them inside one of these list elements. Again, it's all about semantics; placing content in a list makes it clear that you are enumerating a list.

## List

HTML5 supports both an *ordered list* using the `ol` element as well as an *unordered list* using the `ul` element. They work pretty much the same except for a few additional attributes that are available with an ordered list.

With either list, the order of the elements is fixed and defined by the order that the items are included in the list. The appropriate type is chosen based on semantics. If the order of the list is not meaningful, use the unordered list. For example, if you're listing some popular sports such as Football, Baseball, and Basketball, use an unordered list. In contrast, if you are specifying the top three most popular sports, you might want to list them in order of popularity. In this case the order is meaningful and you should use an ordered list.

The default style of an unordered list is to prefix each item with a bullet of some type and the same bullet is used for all of the items. For an ordered list, the default style will use a number, and the numbers will be assigned sequentially.

The items within a list (either ordered or unordered) are represented by a *list item* (`li`) element. The only elements that can be included inside either a `ul` or `ol` element are `li` elements. However, you can put any flow element inside an `li` element. Listing 4-2 demonstrates a simple unordered list and an ordered list. With default styling, this will be rendered as shown in Figure 4-8.

**Listing 4-2.** Sample unordered and ordered lists

```
<h2>Book Topics</h2>
<ul>
  <li>HTML</li>
  <li>CSS</li>
  <li>JavaScript</li>
</ul>
<h2>HTML Chapters</h2>
<ol start="4">
  <li>Structural Elements</li>
  <li>Text Elements</li>
  <li>Table Elements</li>
  <li>Embedded Elements</li>
  <li>Form Elements</li>
</ol>
```

## Book Topics

- HTML
- CSS
- JavaScript

## HTML Chapters

4. Structural Elements
5. Text Elements
6. Table Elements
7. Embedded Elements
8. Form Elements

**Figure 4-8.** Sample unordered and ordered list

As I mentioned, the ordered list supports a few additional attributes. The most common one is the `start` attribute, which I used in the previous example. In this example, I'm listing the chapters of this book that cover the HTML elements. The first three chapters contained introductory material so the first chapter on HTML is this chapter, Chapter 4. The `start` attribute indicates the number to use for the first `li` element; in this case 4. If omitted, the numbering will start with 1.

The `reversed` attribute is a Boolean attribute that is used to assign the numbers in reverse order. You would use this, for example, if you were listing the top three most popular sports and wanted to leave the most popular for last. The numbering would be in reverse order, 3, 2, 1. If you do not specify the `start` attribute, the number of the first item is set so that the last element will be 1. If you put ten items in a list and specify the `reversed` attribute with no `start` attribute, the items will be numbered from 10 to 1. If you do specify `start` along with `reversed`, the first item will use whatever was specified in the `start` attribute and the rest will go down from there, so you could end up with negative numbers.

---

■ **Tip** The `reversed` attribute does not change the order of the `li` elements. They are always rendered in the order they are included in the list. The `reversed` attribute only affects the numbering of the items.

---

The `type` attribute specifies what type of “numbering” to use. Here are the supported values:

- 1 - Use numbers (this is the default value if not specified)
- A - Use uppercase letters (e.g., A, B, C, D)
- a - Use lowercase letters
- I - Use Roman numerals with uppercase letters (e.g., I, II, III, IV)
- i - Use lowercase Roman numerals

The `start` attribute is always numeric regardless of what value is specified for `type`. It will be converted to the appropriate type representation. For example, if you specify 4 and the `type` is I, the first element will use IV.

---

■ **Caution** Letters or Roman numerals do not support zero or negative numbers. If these values are needed, they will be displayed as numbers. For example, if you have five items in the list and specify the `ol` element as `<ol start="3" type="a" reversed>`, the numbering will be c, b, a, 0, -1.

---

You can use a list to present a series of links such as a menu. For example:

```
<nav>
  <h1>Navigation</h1>
  <ul>
    <li><a href="/">Home</a></li>
    <li><a href="http://www.apress.com">Apress</a></li>
    <li><a href="http://www.theCreativePeople.com">My Site</a></li>
  </ul>
</nav>
```

This will display a bulleted list of links. However, with CSS you can render this in a variety of ways including buttons or horizontally arrayed links. I will demonstrate this in Chapter 13.

## Description List

The *description list* (`d1`) element is used to define a list of terms. It is implemented as a set of name/value pairs. It is often used to create a glossary. In this case the names are the terms being defined and the values are the definition or description of those terms. You can put series of term (`dt`) elements and description (`dd`) elements inside the `d1` element.

For example, a simple `d1` element might be used like this:

```
<d1>
  <dt>Term1</dt>
  <dd>Definition</dd>
  <dt>Term2</dt>
  <dd>Definition</dd>
</d1>
```

---

■ **Note** In HTML4, the `d1` element was called a definition list and was intended to define a list of terms. The syntax has not changed in HTML5, but the name was changed to description list to convey a broader use of the element. In HTML5 it is used to provide a grouping of term/description pairs.

---

Listing 4-3 illustrates how this pattern can be used to list a series of books along with a description of each. Figure 4-9 shows how this would be displayed in a browser using only default formatting.

**Listing 4-3.** Using dl, dt, and dd elements

```

<dl>
  <dt>Beginning WF</dt>
  <dd>
    Indexed by feature so you can find answers easily and written in an accessible style,
    Beginning WF shows how Microsoft's Workflow Foundation (WF) technology can be used in a wide
    variety of applications.
  </dd>
  <dt>Office 2010 Workflow</dt>
  <dd>
    Workflow is the glue that binds information worker processes, users, and artifacts—
    without it, information workers are just islands of data and potential. Office 2010 Workflow
    walks you through implementing workflow solutions.
  </dd>
  <dt>Pro Access 2010 Development</dt>
  <dd>
    Pro Access 2010 Development is a fundamental resource for developing business
    applications that take advantage of the features of Access 2010. You'll learn how to build
    database applications, create Web-based databases, develop macros and VBA tools for Access
    applications, integrate Access with SharePoint, and much more.
  </dd>
</dl>

```

**Beginning WF**

Indexed by feature so you can find answers easily and written in an accessible style,  
Beginning WF shows how Microsoft's Workflow Foundation (WF) technology can be  
used in a wide variety of applications.

**Office 2010 Workflow**

Workflow is the glue that binds information worker processes, users, and artifacts—  
without it, information workers are just islands of data and potential. Office 2010  
Workflow walks you through implementing workflow solutions.

**Pro Access 2010 Development**

Pro Access 2010 Development is a fundamental resource for developing business  
applications that take advantage of the features of Access 2010. You'll learn how to build  
database applications, create Web-based databases, develop macros and VBA tools for  
Access applications, integrate Access with SharePoint, and much more.

**Figure 4-9.** Rendering a description list

However, you're not restricted to this one-to-one mapping. Another use of the dl element is to list groups of things with a group header. You accomplish this by using the dt element as the group header and the dd elements as the group members. For example, Listing 4-4 contains the starting lineup for the New York Yankees, organized by position categories. Figure 4-10 shows the default rendering of this content.



**Listing 4-4.** Using multiple description elements for each term

```
<h2>New York Yankees Starting Lineup</h2>
<dl>
  <dt>Infielders</dt>
  <dd>Teixeira, 1B</dd>
  <dd>Castro, 2B</dd>
  <dd>Gregorius, SS</dd>
  <dd>Torreyes, 3B</dd>
  <dd>McCann, C</dd>
  <dt>Outfielders</dt>
  <dd>Hicks, LF</dd>
  <dd>Ellsbury, CF</dd>
  <dd>Ackley, RF</dd>
  <dt>Designated Hitter</dt>
  <dd>Rodriguez, DH</dd>
  <dt>Pitchers</dt>
  <dd>Gausman, R</dd>
  <dd>Wilson, R</dd>
  <dd>Tillman, R</dd>
  <dd>Price, L</dd>
  <dd>Porcello, R</dd>
</dl>
```

## New York Yankees Starting Lineup

Infielders  
 Teixeira, 1B  
 Castro, 2B  
 Gregorius, SS  
 Torreyes, 3B  
 McCann, C  
 Outfielders  
 Hicks, LF  
 Ellsbury, CF  
 Ackley, RF  
 Designated Hitter  
 Rodriguez, DH  
 Pitchers  
 Gausman, R  
 Wilson, R  
 Tillman, R  
 Price, L  
 Porcello, R

**Figure 4-10.** The New York Yankees Starting Lineup

The other possibility is to have multiple terms refer to the same description. You can do this as well but you need to keep the like terms together. For example:

```
<dl>
  <dt>Hicks, LF</dt>
  <dt>Ellsbury, CF</dt>
  <dt>Ackley, RF</dt>
  <dd>Outfielders - the positions in baseball that are played in the grassy area behind
the diamond, known as the outfield. These positions include the Left Fielder, Center
Fielder, and Right Fielder.</dd>
</dl>
```

The term element is intended to be relatively short such as a word or phrase. There are specific limitations as what kind of elements can be included in a `dt` element. You cannot use a header, footer, or any sectioning content inside a `dt` element. Even at that, there's nothing to prevent you from including paragraphs of content inside a `dt` element. That is inappropriate, however, when you consider the semantic rules. A `dl` element is used to describe a list of terms, and a paragraph of text is not a term.

There are no such restrictions on the definition element. You can place any flow content inside a `dd` element, including sectioning and embedded content.

## Inline Frames

The `iframe` element is used to embed another web page within the current document. I'll explain it here briefly, and we'll look at it more closely in Chapter 17. You can include an `iframe` with markup like this:

```
<iframe src="http://www.apress.com" width="100%" height="400">
  <p>Your browser does not support iframes</p>
</iframe>
```

The content inside the `iframe` element will be rendered only if the `iframe` element is not supported by the browser. The `iframe` element supports several attributes that you can use to configure how the element is rendered and what options it will allow.

- `allowfullscreen` - If this Boolean attribute is specified, scripts within the embedded page may switch to fullscreen mode using the `requestFullscreen()` method call.
- `height` - The height of the `iframe` element in pixels or a percentage of the parent page.
- `name` - The name attribute can be used when providing a link to the `iframe` from within the parent document.
- `sandbox` - This can be used to restrict what can be done in the embedded page. This will be covered in Chapter 17.
- `src` - This is used to specify the URL of the page that should be embedded.
- `srcdoc` - If specified and supported, the `srcdoc` attribute will override the value of the `src` attribute. This is normally used in conjunction with the `sandbox` attribute.
- `width` - The width of the `iframe` element in pixels or a percentage of the parent page.

## Deprecated Elements

Some grouping elements have been dropped from the HTML5 specification. I will describe them here briefly. These are likely to be supported for some time to ensure backward compatibility. However, you should not use these for any new development.

### hgroup

The `hgroup` element was used to group the `h1`–`h6` elements and hide all but the first one from the document outline. If you wanted a title for a section as well as a subtitle, for example, you might use an `h1` element for the title and `h2` for the subtitle. However, the `h2` element would start a new section in the document outline. Putting them both into an `hgroup` element would create only a single section, using the `h1` title.

Instead of using a `hgroup` element, the recommendation is to put the title and subtitle(s) into a header element. The title should be in an `h1`–`h6` element to define the section and be added to the document outline. The remaining header content such as subtitles or taglines should be placed in `p` or `span` tags and then styled appropriately using CSS. For more information and examples, see the W3C specification at <http://www.w3.org/TR/html5/common-idioms.html#sub-head>.

### dir

The directory list (`dir`) element was used to present a directory listing. The items within this list are included in an `li` element. The `dir` element has been deprecated in HTML5. You should use the unordered list (`ul`) instead.

### frame and frameset

The `frame` and `frameset` elements were used to organize a page into sections. This includes both logical sections as well as page layout. The logical sectioning is now accomplished using the sectioning content that I described earlier such as `section`, `aside`, and `article`. The layout is configured through CSS, which I will demonstrate in Chapter 13.

## Summary

In this chapter I demonstrated all of the HTML elements that are used to organize content. These provide semantics meaning to the content groupings.

A document is organized using sectioning content to define the larger sections of the document. These along with the heading elements (`h1`–`h6`) define the document outline. The header and footer elements are used to identify introductory and concluding content both at the document level as well as within a section. Sectioning roots including `blockquote`, `details`, and `figure` elements have their own sections that do not affect the document outline.

There are several other grouping elements that do not affect the outline but provide semantic grouping at a lower level. These include the `main`, paragraph (`p`), horizontal rule (`hr`), and division (`div`) elements. The preformatted (`pre`) element is used primarily to preserve the formatting of the content, rather than for semantic reasons. Finally, ordered lists (`ol`), unordered lists (`ul`), and description lists (`dl`) provide ways to present a list of things.

By using these elements, you provide semantic details about your content, which will make it easier to apply consistent styling rules. Even without any custom CSS, the document begins to take “shape” as the default styling renders the content consistently with the semantic purposes of each element.

The HTML document shown in Listing 4-5 demonstrates all of the techniques described in this chapter. This is also available in the source code download. In the next chapter, I’ll start to put some actual content onto this structure. I’ll explain the various elements that will provide semantic meaning to the textual content.

**Listing 4-5.** The complete HTML document for Chapter 4

```

<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="author" content="Mark J Collins" />
    <meta name="description" content="Sample HTML document" />
    <title>HTML5 Sample</title>
    <link rel="stylesheet" href="Initial.css" />
    <link rel="stylesheet" href="Print.css" media="print" />
    <link rel="icon" type="image/x-icon" href="HTMLBadge.ico" />
    <link rel="alternate" type="application/pdf" href="MainPage.pdf" />
    <link rel="author" href="mailto:markc@thecreativepeople.com" />
    <link rel="author" type="text/html" href="http://www.thecreativepeople.com" />
  </head>
  <body>
    <h1>My Sample Page</h1>
    <header>
      <p>Heading</p>
    </header>
    <nav>
      <h1>Navigation</h1>
      <ul>
        <li><a href="/">Home</a></li>
        <li><a href="http://www.apress.com">Apress</a></li>
        <li><a href="http://www.theCreativePeople.com">My Site</a></li>
      </ul>
    </nav>
    <section>
      <h1>Top-level</h1>
      <section>
        <h2>New York Yankees Starting Lineup</h2>
        <details>
          <summary>This content is collapsed</summary>
          <dl>
            <dt>Infielders</dt>
            <dd>Teixeira, 1B</dd>
            <dd>Castro, 2B</dd>
            <dd>Gregorius, SS</dd>
            <dd>Torreyes, 3B</dd>
            <dd>McCann, C</dd>
            <dt>Outfielders</dt>
            <dd>Hicks, LF</dd>
            <dd>Ellsbury, CF</dd>
            <dd>Ackley, RF</dd>
            <dt>Designated Hitter</dt>
            <dd>Rodriguez, DH</dd>
            <dt>Pitchers</dt>
            <dd>Gausman, R</dd>
            <dd>Wilson, R</dd>
            <dd>Tillman, R</dd>
          </dl>
        </details>
      </section>
    </section>
  </body>
</html>

```

```

        <dd>Price, L</dd>
        <dd>Porcello, R</dd>
    </dl>
    <dl>
        <dt>Hicks, LF</dt>
        <dt>Ellsbury, CF</dt>
        <dt>Ackley, RF</dt>
        <dd>Outfielders - the positions in baseball that are played in the
grassy area behind the diamond, known as the outfield. These positions include the Left
Fielder, Center Fielder, and Right Fielder.</dd>
    </dl>
</details>
<main>
    <h1>Main content</h1>
    <section>
        <h1>Featured content</h1>
        <blockquote cite="www.apress.com">
            <h1>Quotation</h1>
            <p>This is a quotation</p>
        </blockquote>
        <p>paragraph 1</p>
        <hr />
        <p>paragraph 2</p>
    </section>
</main>
<section>
    <h1>Articles</h1>
    <article>
        <h1>Article 1</h1>
        <pre>I heard the bells on Christmas Day
Their old, familiar carols play,
    And wild and sweet
    The words repeat
Of peace on earth, good-will to men!</pre>
        <cite>Henry Wadsworth Longfellow - 1863, public domain</cite>
        <figure>
            <h1>Figure</h1>
            
            <figcaption>Official HTML5 Logo</figcaption>
        </figure>
    </article>
</section>
</section>
<aside>
    <h1>Related content</h1>
    <section>
        <h1>HTML Reference</h1>
        <h2>Book Topics</h2>
        <ul>
            <li>HTML</li>
            <li>CSS</li>
            <li>JavaScript</li>
        </ul>
    </section>
</aside>

```

```

        <h2>HTML Chapters</h2>
        <ol start="3" type="I" reversed>
            <li>Structural Elements</li>
            <li>Text Elements</li>
            <li>Table Elements</li>
            <li>Embedded Elements</li>
            <li>Form Elements</li>
        </ol>
    </section>
    <section>
        <h1>Book list</h1>
        <dl>
            <dt>Beginning WF</dt>
            <dd>Indexed by feature so you can find answers easily and written in
an accessible style, Beginning WF shows how Microsoft's Workflow Foundation (WF) technology
can be used in a wide variety of applications.</dd>
            <dt>Office 2010 Workflow</dt>
            <dd>Workflow is the glue that binds information worker processes,
users, and artifacts—without it, information workers are just islands of data and potential.
Office 2010 Workflow walks you through implementing workflow solutions.</dd>
            <dt>Pro Access 2010 Development</dt>
            <dd>Pro Access 2010 Development is a fundamental resource for
developing business applications that take advantage of the features of Access 2010. You'll
learn how to build database applications, create Web-based databases, develop macros and VBA
tools for Access applications, integrate Access with SharePoint, and much more.</dd>
        </dl>
    </article>
    <h1>Book 1</h1>
    <iframe src="http://www.apress.com" width="100%" height="400">
        <p>Your browser does not support iframes</p>
    </iframe>
</article>
</section>
</aside>
</section>
<footer>
    <p>Closing content</p>
    <address>
        <p>Provided by
            <a href="mailto:mcollins@theCreativePeople.com">Mark J. Collins</a>
        </p>
        <p>For more information
            <a href="http://www.theCreativePeople.com">visit his website</a>
        </p>
    </address>
</footer>
</body>
</html>

```