Browsers that support the ins and del elements may give it special visual treatment (for example, displaying deleted text in strike-through text), but authors are encouraged to use style sheets to provide presentational instructions.

The title attribute may be used with del or ins to provide a short explanation for the change that may be displayed as a "tool tip" on visual browsers. The cite attribute provides a way to add links to longer explanations, but it is poorly supported as of this writing.

The datetime attribute may be used to indicate the date and time the change was made (although it, too, is poorly supported). Dates and times follow the format listed above where YYYY is the four-digit year, MM is the two-digit month, DD is the day, hh is the hour (00 through 23), mm is the minute (00 through 59), and ss is the seconds (00 through 59). The TZD stands for Time Zone Designator and its value can be Z (to indicate UTC, Coordinated Universal Time), an indication of the number of hours and minutes ahead of UTC (such as +03:00), or an indication of the number of hours and minutes behind UTC (such as –02:20). This is the standard format for date and time values in HTML. For more information, see *www.w3.org/TR/1998/NOTE-datetime-19980827*.

Generic Elements (div and span)

The generic div and span elements provide a way for authors to create custom elements. The div element is used to indicate block-level elements, while span indicates inline elements. Both generic elements rely on id and class attributes to give them a name, meaning, or context.

The Versatile div

The div element is used to identify and label any block-level division of text, whether it is a few list items or an entire page.

div

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

Attributes

Core (id, class, style, title), *Internationalization, Events*

Deprecated attributes

align="center|left|right"

By marking a section of text as a div and giving it a name using id or class attributes, you are essentially creating a custom HTML element. In this example, a heading and a list are enclosed in a div identified as "sidebar."

```
<div id="sidebar">
    <h1>List of links</h1>
    <ul>
        <li>Resource 1</li>
```

```
            <li>Resource 2</li>
            <li>Resource 3</li>
        </ul>
    </div>
```

Because a `div` is a block-level element, its contents will start on a new line (even text not contained within other block-level elements). Otherwise, `div` elements have no inherent presentation qualities of their own.

The `div` really shines when used in conjunction with Cascading Style Sheets. Once you've marked up and named a `div`, you can apply styles to all of its contents or treat it as a box that can be positioned on the page, for instance, to form a new text column. A `div` may also be called on by script, applet, or other processing by user agents.

The Useful span

Like the `div` element, `span` allows authors to create custom elements. The difference is that span is used for inline elements and does not introduce a line break.

span

```
<span>...</span>
```

Attributes

Core (`id`, `class`, `style`, `title`), *Internationalization, Events*

This is a simple example of a span used to identify a telephone number.

```
Jenny: <span class="telephone">867.5309</span>
```

Markup like this has a number of uses. Most commonly, it is a "hook" that can be used to apply style sheet rules. In this example, all elements labeled as `telephone` may receive the same presentational instructions, such as to be displayed in bold, blue text.

The `span` also gives meaning to an otherwise random string of digits to user agents who know what to do with `telephone` information. This is discussed a bit more in the next section.

Element Identifiers (class and id)

The previous examples show how the `id` and `class` attributes are used to turn generic `div` and `span` elements into elements with specific meanings and uses. It should be pointed out that `class` and `id` attributes may be used with nearly all (X)HTML elements, not just `div` and `span`. This section discusses the `id` and `class` element identifiers and their distinct uses.

id identifier

The `id` attribute is used to give an element a specific and unique name in the document. In the earlier `div` example, `id` was used to label a section of the page as "sidebar." That means there may be no other element with `id="sidebar"` in that

document (although, it is okay if it appears in other documents on the same site). ID values must be unique.

The HTML 4.01 Recommendation specifies the following uses for `id` attribute:

- As a style sheet selector
- As a target anchor for links (with the same functionality as `<a name="">`)
- As a means to access an element from a script
- As the name of a declared object element
- For general purpose processing by user agents, essentially treating the element as data

class identifier

The `class` attribute is used for grouping similar elements. Multiple elements may be assigned the same `class` name, and doing so enables them to be treated similarly.

In the `span` example above, the telephone number was identified as `telephone` with the `class` attribute. This implies that there may be many more telephone numbers in the document. A single style sheet rule could then be used to make them all bold and blue. Changing them all to green requires editing just one line of code. This offers an obvious advantage over changing color one by one with the deprecated `font` element. In addition to being inefficient to maintain, `font` doesn't add any semantic cues for user agents.

According to the HTML 4.01 specification, the `class` attribute may be used:

- As a style sheet selector
- For general purpose processing by user agents

In HTML 4.01, `id` and `class` attributes may be used with all elements except `base`, `basefont`, `head`, `html`, `meta`, `param`, `script`, `style`, and `title`. In XHTML, `id` support has been added to those elements.

Tips on using class

There is a heady exhilaration that comes with the ability to create your own custom elements using `id` and `class`. The `class` attribute in particular is prone to misuse. These tips should provide some basic guidance for keeping your markup clean.

Keep `class` *names meaningful.*

The value of the `class` attribute should provide a semantic description of a `div` or `span`'s content. Choosing names based on the intended presentation of the element—for example, `class="indented"` or `class="bluetext"`—does little toward giving the element meaning and reintroduces presentational information to the document. It is also short-sighted. Consider what happens when, in an inevitable future design change, all elements classified as `bluetext` are rendered in green.