IBM

developerWorks Technical topics XML Technical library

New elements in HTML 5

Structure and semantics

HTML 5 introduces new elements to HTML for the first time since the last millennium. New structural elements include `aside`, `figure`, and `section`. New inline elements include `time`, `meter`, and `progress`. New embedding elements include `video` and `audio`. New interactive elements include `details`, `datagrid`, and `command`.

Share:

Elliotte Rusty Harold is originally from New Orleans, to which he returns periodically in search of a decent bowl of gumbo. However, he resides in the Prospect Heights neighborhood of Brooklyn with his wife Beth and cats Charm (named after the quark) and Marjorie (named after his mother-in-law). He's an adjunct professor of computer science at Polytechnic University, where he teaches Java and object-oriented programming. His Cafe au Lait Web site has become one of the most popular independent Java sites on the Internet, and his spin-off site, Cafe con Leche, has become one of the most popular XML sites. His next book, Refactoring HTML, will be published by Addison Wesley later this year. He's currently working on the XOM API for processing XML and the Jaxen XPath engine.

07 August 2007

Also available in Russian Japanese Vietnamese

Development of HTML stopped in 1999 with HTML 4. The W3C focused its efforts on changing the underlying syntax of HTML from Standard Generalized Markup Language (SGML) to XML, as well as completely new markup languages like Scalable Vector Graphics (SVG), XForms, and MathML. Browser vendors focused on browser features like tabs and RSS readers. Web designers started learning CSS and the JavaScript™ language to build their own applications on top of the existing frameworks using Asynchronous JavaScript + XML (Ajax). But HTML itself grew hardly at all in the next eight years.

Frequently used acronyms

CSS: Cascading Style Sheets
HTML: Hypertext Markup Language
W3C: World Wide Web Consortium
XML: Extensible Markup Language

Recently, the beast came back to life. Three major browser vendors—Apple, Opera, and the Mozilla Foundation—came together as the Web Hypertext Application Technology Working Group (WhatWG) to develop an updated and upgraded version of classic HTML. More recently, the W3C took note of these developments and started its own next-generation HTML effort with many of the same members. Eventually, the two efforts will likely be merged. Although many details remain to be argued over, the outlines of the next version of HTML are becoming clear.

This new version of HTML—usually called HTML 5, although it also goes under the name Web Applications 1.0—would be instantly recognizable to a Web designer frozen in ice in 1999 and thawed today. There are no namespaces or schemas. Elements don't have to be closed. Browsers are forgiving of errors. A `p` is still a `p`, and a `table` is still a `table`.

At the same time, this proverbial unfrozen caveman Web designer would encounter some new and confusing elements. Yes, old friends like `div` remain, but now HTML includes `section`, `header`, `footer`, and `nav` as well. `em`, `code`, and `strong` are still present, but so are `meter`, `time`, and `m`. `img` and `embed` continue to be used, but now there are `video` and `audio` too. However, closer inspection by

the caveman designer would reveal that these elements aren't that different. Many of them might be things the designer needed back in 1999 but didn't have. All these new elements are easily learned by simple analogy with elements the designer already understands. In fact, they're a lot easier to learn than Ajax or CSS.

Finally, when the caveman fired up the 300MHz laptop running Windows 98 that was also frozen in 1999, they might be astonished to realize that the new pages display fine in Netscape 4 and Windows® Internet Explorer® 5. Sure, the browser wouldn't recognize or do anything with the new elements, but the page still displays, and the content is all there.

That's not a happy coincidence. HTML 5 was explicitly designed to degrade gracefully in browsers that don't support it. The reason is simple: We are all cave people. Browsers now have tabs, CSS, and XmlHttpRequest, but their HTML renderers are stuck in 1999. The Web can't move forward without accounting for the installed base. HTML 5 understands this. It offers real benefits to page authors today while promising even more to page readers tomorrow as browsers are slowly upgraded. With that in mind, let's look at what HTML 5 brings you.

Structure

Even well-formed HTML pages are harder to process than they should be because of the lack of structure. You have to figure out where the section breaks go by analyzing header levels. Sidebars, footers, headers, navigation menus, main content sections, and individual stories are marked up by the catch-all `div` element. HTML 5 adds new elements to specifically identify each of these common constructs:

`section`: A part or chapter in a book, a section in a chapter, or essentially anything that has its own heading in HTML 4

`header`: The page header shown on the page; not the same as the `head` element

`footer`: The page footer where the fine print goes; the signature in an e-mail message

`nav`: A collection of links to other pages

`article`: An independent entry in a blog, magazine, compendium, and so forth

For example, consider a typical blog front page with a header at the top, a footer at the bottom, several entries, a navigation section, and a sidebar, as shown in Listing 1.

Listing 1. A typical blog page today

```
<html>
  <head>
    <title>Mokka mit Schlag </title>
 </head>
<body>
<div id="page">
  <div id="header">
    <h1><a href="http://www.elharo.com/blog">Mokka mit Schlag</a></h1>
  </div>
  <div id="container">
    <div id="center" class="column">
      <div class="post" id="post-1000572">
        <h2><a href=
      "/blog/birding/2007/04/23/spring-comes-and-goes-in-sussex-county/">
      Spring Comes (and Goes) in Sussex County</a></h2>

        <div class="entry">
          <p>Yesterday I joined the Brooklyn Bird Club for our
          annual trip to Western New Jersey, specifically Hyper
          Humus, a relatively recently discovered hot spot. It
          started out as a nice winter morning when we arrived
          at the site at 7:30 A.M., progressed to Spring around
          10:00 A.M., and reached early summer by 10:15. </p>
```

```
        </div>
      </div>


      <div class="post" id="post-1000571">
        <h2><a href=
          "/blog/birding/2007/04/23/but-does-it-count-for-your-life-list/">
           But does it count for your life list?</a></h2>

        <div class="entry">
          <p>Seems you can now go <a
          href="http://www.wired.com/science/discoveries/news/
          2007/04/cone_sf">bird watching via the Internet</a>. I
          haven't been able to test it out yet (20 user
          limit apparently) but this is certainly cool.
          Personally, I can't imagine it replacing
          actually being out in the field by any small amount.
          On the other hand, I've always found it quite
          sad to meet senior birders who are no longer able to
          hold binoculars steady or get to the park. I can
          imagine this might be of some interest to them. At
          least one elderly birder did a big year on TV, after
          he could no longer get out so much. This certainly
          tops that.</p>
        </div>
      </div>

      </div>

    <div class="navigation">
      <div class="alignleft">
         <a href="/blog/page/2/">&laquo; Previous Entries</a>
       </div>
      <div class="alignright"></div>
    </div>
  </div>

  <div id="right" class="column">
    <ul id="sidebar">
      <li><h2>Info</h2>
      <ul>
        <li><a href="/blog/comment-policy/">Comment Policy</a></li>
        <li><a href="/blog/todo-list/">Todo List</a></li>
      </ul></li>
      <li><h2>Archives</h2>
        <ul>
          <li><a href='/blog/2007/04/'>April 2007</a></li>
          <li><a href='/blog/2007/03/'>March 2007</a></li>
          <li><a href='/blog/2007/02/'>February 2007</a></li>
          <li><a href='/blog/2007/01/'>January 2007</a></li>
        </ul>
      </li>
    </ul>
  </div>
  <div id="footer">
    <p>Copyright 2007 Elliotte Rusty Harold</p>
  </div>
</div>

</body>
</html>
```

Even with proper indenting, it's a fairly confusing mass of nested `div`s. In HTML 5, you can replace these with more direct semantic elements, as shown in Listing 2.

Listing 2. A typical blog page in HTML 5

```
<html>
 <head>
    <title>Mokka mit Schlag </title>
 </head>
<body>
  <header>
    <h1><a href="http://www.elharo.com/blog">Mokka mit Schlag</a></h1>
  </header>
  <section>
      <article>
        <h2><a href=
        "/blog/birding/2007/04/23/spring-comes-and-goes-in-sussex-county/">
         Spring Comes (and Goes) in Sussex County</a></h2>

        <p>Yesterday I joined the Brooklyn Bird Club for our
        annual trip to Western New Jersey, specifically Hyper
        Humus, a relatively recently discovered hot spot. It
        started out as a nice winter morning when we arrived at
        the site at 7:30 A.M., progressed to Spring around 10:00
        A.M., and reached early summer by 10:15. </p>
      </article>


      <article>
        <h2><a href=
          "/blog/birding/2007/04/23/but-does-it-count-for-your-life-list/">
          But does it count for your life list?</a></h2>

          <p>Seems you can now go <a
```

```
          href="http://www.wired.com/science/discoveries/news/
          2007/04/cone_sf">bird watching via the Internet</a>. I
          haven't been able to test it out yet (20 user
          limit apparently) but this is certainly cool.
          Personally, I can't imagine it replacing
          actually being out in the field by any small amount.
          On the other hand, I've always found it quite
          sad to meet senior birders who are no longer able to
          hold binoculars steady or get to the park. I can
          imagine this might be of some interest to them. At
          least one elderly birder did a big year on TV, after
          he could no longer get out so much. This certainly
          tops that.</p>
      </article>
    <nav>
      <a href="/blog/page/2/">&laquo; Previous Entries</a>
    </nav>
  </section>

  <nav>
    <ul>
      <li><h2>Info</h2>
      <ul>
        <li><a href="/blog/comment-policy/">Comment Policy</a></li>
        <li><a href="/blog/todo-list/">Todo List</a></li>
      </ul></li>
      <li><h2>Archives</h2>
        <ul>
          <li><a href='/blog/2007/04/'>April 2007</a></li>
          <li><a href='/blog/2007/03/'>March 2007</a></li>
          <li><a href='/blog/2007/02/'>February 2007</a></li>
          <li><a href='/blog/2007/01/'>January 2007</a></li>
        </ul>
      </li>
    </ul>
  </nav>
  <footer>
    <p>Copyright 2007 Elliotte Rusty Harold</p>
  </footer>

</body>
</html>
```

No `div`s are needed any more. Rather than using site-specific class attributes, the meaning of the different sections can be inferred from standard names. This is especially important for audio, cell-phone, and other nonstandard browsers.

Block semantic elements

As well as the structural elements, HTML 5 adds some purely semantic block-level elements:

`aside`

`figure`

`dialog`

I use the first two all the time in articles like this one and in my books. The third I don't use so much myself, but it's common in much written text.

aside

The `aside` element represents a note, a tip, a sidebar, a pullquote, a parenthetical remark, or something that's just outside the main flow of the narrative. For example, in developerWorks articles, you often find sidebars encoded as tables, as shown in Listing 3.

Listing 3. A developerWorks HTML 4 sidebar

```
<table align="right" border="0" cellpadding="0" cellspacing="0" width="40%">
<tbody><tr><td width="10">
<img alt="" src="//www.ibm.com/i/c.gif" height="1" width="10"></td>
<td>
<table border="1" cellpadding="5" cellspacing="0" width="100%">
<tbody><tr><td bgcolor="#eeeeee">
<p><a name="xf-value"><span class="smalltitle">.xf-value</span></a></p>
<p>
The <code type="inline">.xf-value</code> selector used here styles the input
field value but not its label. This is actually inconsistent
with the current CSS3 draft. The example really should use the
<code type="inline">::value</code> pseudo-class instead like so:
</p>
```

```
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tbody><tr><td class="code-outline">
<pre class="displaycode">input::value { width: 20em; }
#ccnumber::value { width: 18em }
#zip::value { width: 12em }
#state::value { width: 3em  }</pre>
</td></tr></tbody></table><br>

<p>
However, Firefox doesn't yet support this syntax.
</p>
</td></tr></table>
```

In HTML 5, you can write this much more sensibly, as shown in Listing 4.

Listing 4. A developerWorks HTML 5 sidebar

```
<aside>
<h3>.xf-value</h3>
<p>
The <code type="inline">.xf-value</code> selector used here styles the input
field value but not its label. This is actually inconsistent
with the current CSS3 draft. The example really should use the
<code type="inline">::value</code> pseudo-class instead like so:
</p>

<pre class="displaycode">input::value { width: 20em; }
#ccnumber::value { width: 18em }
#zip::value { width: 12em }
#state::value { width: 3em  }</pre>

<p>
However, Firefox doesn't yet support this syntax.
</p>
</aside>
```

The browser can figure out where to put the sidebar, possibly with a little help from CSS.

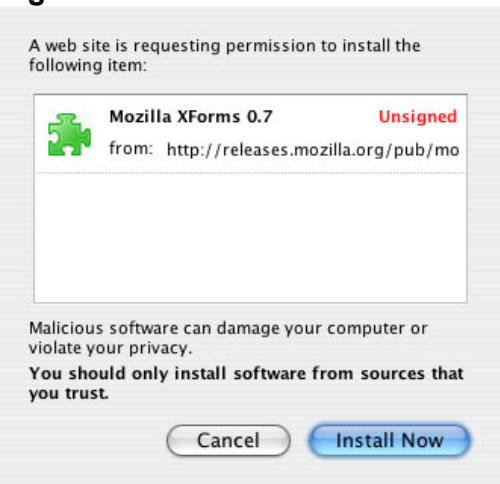
figure

The `figure` element represents a block-level image, along with a caption. For example, in many developerWorks articles, you find markup like Listing 5; the results are shown in Figure 1.

Listing 5. A developerWorks HTML 4 figure

```
<a name="fig2"><b>Figure 2. Install Mozilla XForms dialog</b></a><br />
<img alt="A Web site is requesting permission to install the following item:
   Mozilla XForms 0.7 Unsigned"
  src="installdialog.jpg" border="0" height="317" hspace="5" vspace="5" width="331" />
<br />
```

Figure 1. Install Mozilla XForms dialog



In HTML 5, you can write this more semantically, as shown in Listing 6.

Listing 6. A developerWorks HTML 5 figure

```
<figure id="fig2">
  <legend>Figure 2. Install Mozilla XForms dialog</legend>
  <img alt="A Web site is requesting permission to install the following item:
    Mozilla XForms 0.7 Unsigned"
    src="installdialog.jpg" border="0" height="317" hspace="5" vspace="5" width="331" />
</figure>
```

Most important, browsers—especially screen readers—can clearly and unambiguously associate the

caption with the picture.

The `figure` element isn't limited to pictures. You can also use it to caption `audio`, `video`, `iframe`, `object`, and `embed` elements.

dialog

The `dialog` element represents a conversation between several people. The HTML 5 `dt` element is overloaded to indicate the speaker, and the HTML 5 `dd` element is overloaded to indicate the speech. This gives reasonable display even in legacy browsers. Listing 7 shows a bit of famous dialogue from Galileo's "Dialogue Concerning the Two Chief World Systems."

Listing 7. A Galilean dialogue in HTML 5

```
<dialog>
        <dt>Simplicius </dt>
    <dd>According to the straight line AF,
        and not according to the curve, such being already excluded
        for such a use.</dd>

        <dt>Sagredo </dt>
    <dd>But I should take neither of them,
        seeing that the straight line AF runs obliquely. I should
        draw a line perpendicular to CD, for this would seem to me
        to be the shortest, as well as being unique among the
        infinite number of longer and unequal ones which may be
        drawn from the point A to every other point of the opposite
        line CD. </dd>

        <dt>Salviati </dt>
    <dd><p> Your choice and the reason you
        adduce for it seem to me most excellent. So now we have it
        that the first dimension is determined by a straight line;
        the second (namely, breadth) by another straight line, and
        not only straight, but at right angles to that which
        determines the length. Thus we have defined the two
        dimensions of a surface; that is, length and breadth. </p>

        <p> But suppose you had to determine a height—for
        example, how high this platform is from the pavement down
        below there. Seeing that from any point in the platform we
        may draw infinite lines, curved or straight, and all of
        different lengths, to the infinite points of the pavement
        below, which of all these lines would you make use of? </p>
        </dd>
</dialog>
```

The exact syntax of this element is still being argued over. Some people want to embed additional nondialogue text (such as stage directions) inside the `dialog` element, and others aren't happy with the overloading of `dt` and `dd`. However, most everyone agrees that some such semantic representation of dialogue is a good thing, even if they haven't yet agreed on the exact syntax.

Inline semantic elements

HTML 4 has five different inline elements to represent subtly different variations of computer code: `var`, `code`, `kbd`, `tt`, and `samp`. However, it doesn't have any way to indicate such basic qualities as time, numbers, or sarcasm. HTML 5 aims to rectify this imbalance between techies and normal writers with several new inline elements.

mark

The `m` element indicates text that is "marked" somehow but not necessarily emphasized. You can imagine it as being like highlighted passages in a book. The canonical use case is Google's cached pages. When you follow a link to the cached copy, the search terms are marked. For example, if you searched for "Egret", then a cached Google page might be marked up like this:

```
The Great <m>Egret</m> (also known as the
American <m>Egret</m>)  is a large white wading bird found worldwide.
The Great <m>Egret</m> flies with slow wing beats. The
scientific name of the Great <m>Egret</m> is <i>Casmerodius
albus</i>.
```

The name of this element is currently subject to some debate. It might be changed to `mark` instead of `m` before the spec is released.

time

The `time` element indicates a specific moment in history, such as 5:35 P.M., EST, April 23, 2007. For example,

```
<p>I am writing this example at
<time>5:35 P.M. on April 23rd</time>.
</p>
```

The `time` element helps browsers and others recognize times in HTML pages. It doesn't require any particular format for the element's content. However, each `time` element should have a `datetime` attribute that includes the time in a more machine-recognizable form, like this:

```
<p>I am writing this example at
<time datetime="2007-04-23T17:35:00-05:00">5:35 P.M. on April 23rd</time>.
</p>
```

Machine-readable times are potentially useful for search engines, calendar programs, and the like.

meter

The `meter` element represents a numeric value in a specified range. For example, you can use it for salaries, percentage of the French electorate that voted for Le Pen, or test scores. Here, I use `meter` to mark up some data I got from a Google programmer at Software Development 2007:

```
<p>An entry level programmer in Silicon Valley
can expect to start around <meter>$90,000</meter> per year.
</p>
```

The `meter` element helps browsers and other clients recognize *amounts* in HTML pages. It doesn't require any particular format for the element's content. However, each `meter` element can have up to six attributes offering information about this amount in a more machine-recognizable form:

`value`

`min`

`low`

`high`

`max`

`optimum`

Each of these should contain a decimal number indicating the relevant range. For example, a final exam grade might be marked up like this:

```
<p>Your score was
<meter value="88.7" min="0" max="100" low="65" high="96" optimum="100">B+</meter>.
</p>
```

This indicates that the student's score was 88.7 out of a possible 100. The lowest possible grade was 0, but the lowest actual grade anyone got was 65. The highest grade anyone got was 96, although of course the ideal score was 100. User agents can display this information using some sort of meter control or give the extra data in a tooltip, but most will probably style it like any other inline element.

progress

The `progress` element represents the state of an ongoing process, like the progress bar in a graphical user interface (GUI) application. For instance, it can show you what percentage of a file is downloaded or how far you are into a movie. This progress control says that a download is 33% complete:

```
<p>Downloaded:
  <progress value="1534602" max="4603807">33%</progress>
</p>
```

The `value` attribute shows the current state of the operation. The `max` attribute shows the total amount toward which the progress is moving. Here the element indicates that 1,534,602 bytes out of a total 4,603,807 bytes have been downloaded.

You can display indefinite progress bars by omitting the `max` attribute.

You should use the JavaScript language to dynamically update the progress bar as the operation continues. Statically, this element isn't very interesting.

Embedded media

Video on the Web is booming, but it's almost all proprietary. YouTube uses Flash, Microsoft uses Windows Media®, and Apple uses QuickTime. Markup that works for embedding such content in one browser doesn't work in the next. Consequently, the WhatWG has proposed a new `video` element that allows the embedding of arbitrary video formats. For example, I might embed my QuickTime movie of a Sora in Prospect Park like so:

```
<video src="http://www.cafeaulait.org/birds/sora.mov" />
```

Whether any one format and codec will be preferred is still under debate. Probably Ogg Theora support at least will be strongly recommended, if not required. Support for proprietary formats such as QuickTime and patent-encumbered formats such as MPEG-4 will be optional. Most likely, the actual formats will be decided in the marketplace, much as GIF, JPEG, and PNG became the preferred formats for `img` elements over contenders like BMP, X-Bitmap, and JPEG 2000.

A complementary `audio` element is also proposed. For example,you might attach background music to a Web page like this:

```
<audio src="spacemusic.mp3"
    autoplay="autoplay" loop="20000" />
```

The `autoplay` attribute tells the browser to begin playing as soon as the page is loaded, without waiting for an explicit user request. It then loops 20,000 times before shutting off (or until the user closes the window or goes to another page). Of course, browsers can and should offer users the ability to mute and pause embedded media, whether the page author has done so or not.

Browsers must support the WAV format. Browsers can also support other formats such as MP3 if they like.

Because these elements aren't supported by legacy browsers and can be inaccessible to blind and deaf users, the `audio` and `video` elements might contain additional markup describing the content of the audio and video. This also helps search engines. Ideally these would be full transcripts of the content of the audio and video. For example, Listing 8 shows how you might mark up John F. Kennedy's inaugural address.

Listing 8. John F. Kennedy's inaugural address in HTML 5

```
<audio src="kennedyinauguraladdrees.mp3">
    <p>
    Vice President Johnson, Mr. Speaker, Mr. Chief Justice,
    President Eisenhower, Vice President Nixon, President Truman,
    Reverend Clergy, fellow citizens:
    </p>

    <p>
    We observe today not a victory of party, but a celebration of
    freedom -- symbolizing an end, as well as a beginning --
   signifying renewal, as well as change. For I have sworn before
    you and Almighty God the same solemn oath our forebears
    prescribed nearly a century and three-quarters ago.
    </p>

    <p>
    The world is very different now. For man holds in his mortal
    hands the power to abolish all forms of human poverty and all
    forms of human life. And yet the same revolutionary beliefs for
    which our forebears fought are still at issue around the globe --
    the belief that the rights of man come not from the
    generosity of the state, but from the hand of God.
    </p>

    <p>
    ...
    </p>

    </audio>
```

Interactivity

HTML 5 also goes under the rubric of Web Applications 1.0. Toward that end, several new elements are focused on more interactive experiences for Web pages:

`details`

`datagrid`

`menu`

`command`

These elements all have the potential to change what is displayed based on user action and choice without loading a new page from the server.

details

The `details` element represents further information that might not be shown by default. An optional `legend` element can summarize the details. One use for details is for footnotes and endnotes. For example:

```
The bill of a Craveri's Murrelet is about 10% thinner
than the bill of a Xantus's Murrelet.
<details>
<legend>[Sibley, 2000]</legend>
<p>Sibley, David Allen, The Sibley Guide to Birds,
(New York: Chanticleer Press, 2000) p. 247
 </p>
</details>
```

The exact rendering isn't specified. One browser might use a footnote, another an endnote, and a third a tooltip.

Each `details` element can have an `open` attribute. If it has this attribute, then the details will be initially shown to the reader. If it doesn't have such an attribute, then they will be hidden until the user asks for them. In either case, the user can click an icon or other indicator to show or hide the details.

datagrid

The `datagrid` element serves the role of a grid control. It's intended for trees, lists, and tables that can be updated by both the user and scripts. By contrast, traditional tables are mostly intended for static data.

A `datagrid` gets its initial data from its contents: a `table`, `select`, or other group of HTML elements. For example, Listing 9 shows a `datagrid` that contains a grade sheet. In this example, the datagrid is populated from a `table`. A simpler one-dimensional `datagrid` might be populated by a `select` element. If other HTML elements are used, then each child element becomes a row in the grid.

Listing 9. A grade sheet datagrid

```
<datagrid>
  <table>
    <tr><td>Jones</td><td>Allison</td><td>A-</td><td>B+</td><td>A</td></tr>
    <tr><td>Smith</td><td>Johnny</td><td>A</td><td>C+</td><td>A</td></tr>
    <tr><td>Willis</td><td>Sydney</td><td>C-</td><td>D</td><td>F</td></tr>
    <tr><td>Wilson</td><td>Frank</td><td>B-</td><td>B+</td><td>A</td></tr>
  </table>
</datagrid>
```

What distinguishes this from a regular table is that the user can select rows, columns, and cells; collapse rows, columns, and cells; edit cells; delete rows, columns, and cells; sort the grid; and otherwise interact with the data directly in the browser on the client. The JavaScript code monitors the updates. The `HTMLDataGridElement` (Listing 10) interface is added to the Document Object Model (DOM) to support this.

Listing 10. HTMLDataGridElement

```
interface HTMLDataGridElement : HTMLElement {
           attribute DataGridDataProvider data;
  readonly attribute DataGridSelection selection;
           attribute boolean multiple;
           attribute boolean disabled;
  void updateEverything();
  void updateRowsChanged(in RowSpecification row, in unsigned long count);
  void updateRowsInserted(in RowSpecification row, in unsigned long count);
  void updateRowsRemoved(in RowSpecification row, in unsigned long count);
  void updateRowChanged(in RowSpecification row);
  void updateColumnChanged(in unsigned long column);
  void updateCellChanged(in RowSpecification row, in unsigned long column);
};
```

The DOM can also be used to load data into the grid dynamically. That is, the datagrid doesn't have to have children that provide the initial data. Instead, it can be set with `DataGridDataProvider` object, as shown in Listing 11. This enables you to load data from databases, XmlHttpRequest, or anything else JavaScript code can talk to.

Listing 11. DataGridDataProvider

```
interface DataGridDataProvider {
  void initialize(in HTMLDataGridElement datagrid);
  unsigned long getRowCount(in RowSpecification row);
  unsigned long getChildAtPosition(in RowSpecification parentRow,
      in unsigned long position);
  unsigned long getColumnCount();
  DOMString getCaptionText(in unsigned long column);
  void getCaptionClasses(in unsigned long column, in DOMTokenList classes);
  DOMString getRowImage(in RowSpecification row);
  HTMLMenuElement getRowMenu(in RowSpecification row);
  void getRowClasses(in RowSpecification row, in DOMTokenList classes);
  DOMString getCellData(in RowSpecification row, in unsigned long column);
  void getCellClasses(in RowSpecification row, in unsigned long column,
      in DOMTokenList classes);
  void toggleColumnSortState(in unsigned long column);
  void setCellCheckedState(in RowSpecification row, in unsigned long column,
      in long state);
  void cycleCell(in RowSpecification row, in unsigned long column);
  void editCell(in RowSpecification row, in unsigned long column, in DOMString data);
};
```

menu and command

The `menu` element has actually been present in HTML since at least version 2. It was deprecated in HTML 4, but it comes roaring back with new significance in HTML 5. In HTML 5, a `menu` contains `command` elements, each of which causes an immediate action. For example, Listing 12 is a menu that pops up alerts.

Listing 12. HTML 5 menu

```
<menu>
    <command onclick="alert('first command')"  label="Do 1st Command"/>
    <command onclick="alert('second command')" label="Do 2nd Command"/>
    <command onclick="alert('third command')"  label="Do 3rd Command"/>
</menu>
```

Commands can also be turned into check boxes with a `checked="checked"` attribute. You can turn check boxes into radio buttons by specifying a `radiogroup` attribute whose value is the name of the group of mutually exclusive buttons.

In addition to simple lists of commands, you can use the `menu` element to create a toolbar or pop-up context menu by setting the `type` attribute to `toolbar` or `popup`. For example, Listing 13 shows a toolbar such as you might find in a blog editor like WordPress. It uses the `icon` attribute to link to button pictures.

Listing 13. HTML 5 toolbar

```
<menu type="toolbar">
    <command onclick="insertTag(buttons, 0);"  label="strong" icon="bold.gif"/>
    <command onclick="insertTag(buttons, 1);"  label="em" icon="italic.gif"/>
    <command onclick="insertLink(buttons, 2);" label="link" icon="link.gif"/>
    <command onclick="insertTag(buttons, 3);"  label="b-quote" icon="blockquote.gif"/>
    <command onclick="insertTag(buttons, 4);"  label="del" icon="del.gif"/>
    <command onclick="insertTag(buttons, 5);"  label="ins" icon="insert.gif"/>
    <command onclick="insertImage(buttons);"   label="img" icon="image.gif"/>
    <command onclick="insertTag(buttons, 7);"  label="ul" icon="bullet.gif"/>
    <command onclick="insertTag(buttons, 8);"  label="ol" icon="number.gif"/>
    <command onclick="insertTag(buttons, 9);"  label="li" icon="item.gif"/>
    <command onclick="insertTag(buttons, 10);" label="code" icon="code.gif"/>
    <command onclick="insertTag(buttons, 11);" label="cite" icon="cite.gif"/>
    <command onclick="insertTag(buttons, 12);" label="abbr" icon="abbr.gif"/>
    <command onclick="insertTag(buttons, 13);" label="acronym" icon="acronym.gif"/>
</menu>
```

The `label` attribute gives a title for the menu. For example, Listing 14 might be an Edit menu.

Listing 14. HTML 5 Edit menu

```
<menu type="popup" label="Edit">
    <command onclick="undo()"   label="Undo"/>
    <command onclick="redo()"   label="Redo"/>
    <command onclick="cut()"    label="Cut"/>
    <command onclick="copy()"   label="Copy"/>
    <command onclick="paste()"  label="Paste"/>
    <command onclick="delete()" label="Clear"/>
</menu>
```

Menus can be nested inside other menus to create hierarchical menus.

Conclusion

HTML 5 is part of the future of the Web. Its new elements enable clearer, simpler markup that makes pages more obvious. `Div` and `span` still have their places, but those places are much more restricted than they used to be. Many pages will no longer need to use them.

Although not all browsers will support these new elements at first, the same has been true for most elements introduced after HTML was first invented: `img`, `table`, `object`, and many more. Support will come with time. In the meantime, HTML's must-ignore behavior for unrecognized elements means that users with legacy browsers will still be able to read HTML 5 pages. They can do so today. Users with more modern browsers will get an enhanced experience, but no one will be left out.

Eight years is a long time to wait for new features, especially in the fast-moving world of the Web. HTML 5 restores some of the excitement of the early days when Netscape, Microsoft, and others were introducing new elements every other week. At the same time, it takes a much more careful approach to defining these elements so that everyone can use them interoperably. The future looks bright.

Resources

Dig deeper into XML on

Learn

The future of HTML, Part 1: WHATWG (Edd Dumbill, developerWorks, December 2005): Explore the various paths forward for HTML that developers, designers, authors, vendors, standards bodies, and others have proposed.

The future of HTML, Part 2: XHTML 2.0 (Edd Dumbill, developerWorks, January 2006): Examine the next-generation version of Extensible Hypertext Markup Language (XHTML) and the W3C's response to the demand for rich client behavior embodied in Ajax.

XHTML 1.0: Marking up a new dawn (Molly Holzschlag, developerWorks, January 2006): Read about the well-formedness and validity requirements of XHTML 1.0.

WhatWG and HTML 5 FAQ: Find answers to many common questions about what the WhatWG is doing and why.

Web Applications 1.0: Review the current working draft of the HTML 5 specification.

Dialogue Concerning the Two Chief World Systems: Read this version, as translated by Stillman Drake and annotated and condensed by S. E. Sciortino.

Ogg Theora: Try the next-generation open video format.

IBM XML certification: Find out how you can become an IBM Certified Developer in XML and related technologies.

XML and XML Schema: Visit the developerWorks XML Zone for a wide range of technical articles and tips, tutorials, standards, and IBM Redbooks.

developerWorks technical events and Web casts: Stay current with the technology in these sessions.

Get products and technologies

IBM trial software: Build your next development project with trial software available for download directly from developerWorks.

Discuss

Participate in the discussion forum.

developerWorks XML zone: Share your thoughts: After you read this article, post your comments and thoughts in this forum. The XML zone editors moderate the forum and welcome your input.

XML zone discussion forums: Participate in a variety of XML-centered forums.

developerWorks blogs: Get involved in the developerWorks community.