Extracted from:

HTML5 and CSS3, Second Edition

Level Up with Today’s Web Technologies

The Pragmatic Bookshelf

Dallas, Texas • Raleigh, North Carolina

The Pragmatic Programmers

HTML5 and CSS3

Second Edition

Level Up with Today's Web Technologies

Brian P. Hogan

Edited by Susannah Davidson Pfalzer

The team that produced this book includes:

Susannah Davidson Pfalzer (editor)
Potomac Indexing, LLC (indexer)
Candace Cunningham (copyeditor)
David J Kelly (typesetter)
Janet Furlow (producer)
Juliet Benda (rights)
Ellie Callahan (support)

Printed in the United States of America.
ISBN-13: 978-1-937785-59-8
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—October 2013

Tip 16

Improving Table Accessibility

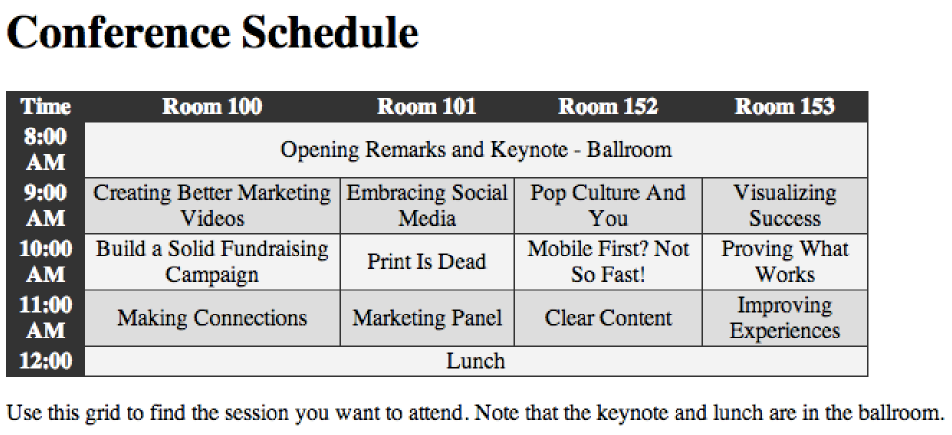
For years, HTML tables have been a great source of pain when it comes to accessibility. It's easy for sighted people to glance at a table and get the context. It can be much more difficult for people using screen readers to understand the big picture. To make matters worse, before CSS let us lay out our content, developers used tables to define the various regions of the page. This created huge problems for screen-reading software because it had to navigate around the tables and figure out how to read them. Unfortunately, even today, some websites rely on tables for layouts, prompting the HTML5 specification to create a special ARIA role for a layout table:

```
➤ <table role="presentation">
    ...
  </table>
```

Even though controlling a page's layout with tables is a horrible practice because it mixes presentation and content, the truth of the matter is that because people have used tables for layout so much, screen-reading software has gotten pretty good about navigating around them. This presentation role helps things out.

Despite the fact that this new role exists, tables aren't for layout. They're designed to let us mark up tabular data, and depending on the complexity of the table, we may need to help the screen readers give the site's visitor some more context. We'll do that by making associations between headers and their associated rows and columns more clear, and we'll add a caption and a description to the table.

AwesomeCo is holding its annual conference, AwesomeConf, in late December, and one of the pages on the site displays the conference schedule for the event, using an HTML table. We've been asked to ensure that this table is readable by screen readers, because in the past some attendees complained on the end-of-conference survey that the site had accessibility issues. The following figure shows the current conference schedule, displayed as an HTML table.



Conference Schedule

Time	Room 100	Room 101	Room 152	Room 153
8:00 AM	Opening Remarks and Keynote - Ballroom			
9:00 AM	Creating Better Marketing Videos	Embracing Social Media	Pop Culture And You	Visualizing Success
10:00 AM	Build a Solid Fundraising Campaign	Print Is Dead	Mobile First? Not So Fast!	Proving What Works
11:00 AM	Making Connections	Marketing Panel	Clear Content	Improving Experiences
12:00	Lunch			

Use this grid to find the session you want to attend. Note that the keynote and lunch are in the ballroom.

Figure 16—AwesomeConf schedule page

Here’s a snippet of the code from the current page.

html5_accessible_tables/original_index.html

```
<h1>Conference Schedule</h1>

<table>
  <tr>
    <th>Time</th>
    <th>Room 100</th>
    <th>Room 101</th>
    <th>Room 152</th>
    <th>Room 153</th>
  </tr>
  <tr>
    <th>8:00 AM</th>
    <td colspan="4">Opening Remarks and Keynote  - Ballroom</td>
  </tr>
  <tr>
    <th>9:00 AM</th>
    <td>Creating Better Marketing Videos</td>
    <td>Embracing Social Media</td>
    <td>Pop Culture And You</td>
    <td>Visualizing Success</td>
  </tr>
</table>
<section>
  <p>
    Use this grid to find the session you want
    to attend.  Note that the keynote and lunch
    are in the ballroom.
  </p>
</section>
```

It's a pretty standard table, but it has headings on both the x- and y-axes. This can present a problem for some screen reader–and-browser combinations. Let's make the heading associations more clear, both in our code and to screen readers.

Associating Headings with Columns

For simple tables, the <th> tag is enough to denote a header. The browsers and screen readers use a somewhat complex algorithm to locate the associated row or column. In more complex tables, we can use the scope attribute to explicitly state that a heading is for a column or a row. Here's how:

html5_accessible_tables/accessible_index.html
```
  <tr>
➤   <th scope="col">Time</th>
➤   <th scope="col">Room 100</th>
➤   <th scope="col">Room 101</th>
➤   <th scope="col">Room 152</th>
➤   <th scope="col">Room 153</th>
  </tr>

  <tr>
➤   <th scope="row">8:00 AM</th>
    <td colspan="4">Opening Remarks and Keynote  - Ballroom</td>
  </tr>

  <tr>
➤   <th scope="row">9:00 AM</th>
    <td>Creating Better Marketing Videos</td>
    <td>Embracing Social Media</td>
    <td>Pop Culture And You</td>
    <td>Visualizing Success</td>
  </tr>
```

For all of the column headings, we specify scope="col". For the row headings, we use scope="row". This makes it easier for screen readers to associate columns, but we can also improve the overall accessibility of the table by describing more clearly what it does.

Explaining Tables with Captions and Descriptions

If we're presenting a table of information, it's a good idea to use some kind of heading or title above or below the table to explain what it does. By putting the title of the table inside a <caption> tag, we allow screen readers to use this to announce the table more clearly. We place the <caption> tag right below the opening <table> tag, like this:

Joe asks:
What About the id and <headers> Attributes?

For many years, it was considered a best practice to associate table headers to columns by assigning a unique id to each header, and then referencing that id in each table cell using the <headers> attribute, like this:

```
<table>
  <tr>
    <th id="name">Name</th>
    <th id="email"></th>
  </tr>
  <tr>
    <td headers="name">Ted</td>
    <td headers="email">ted@puzzlesthebar.com</td>
  </tr>
  <tr>
    <td headers="name">Barney</td>
    <td headers="email">barney@puzzlesthebar.com</td>
  </tr>
</table>
```

For simple tables with lots of rows of data, this approach vastly increases the markup on the page without adding any benefits over using scope. This should be reserved for incredibly complex tables, such as those with nested headers. And if you have tables that are that complex, you should see if you can restructure the information in a more understandable manner.

html5_accessible_tables/accessible_index.html

```
➤ <caption>
➤   <h1>Conference Schedule</h1>
➤ </caption>
  <tr>
```

Sometimes a caption isn’t enough to explain what’s going on with the table. We can use the aria-describedby role to link a table to a section of descriptive content on the page. Our table has a nice descriptive block of text already set aside in a <section> tag. Let’s add an id attribute to that section:

```
➤ <section id="schedule_instructions">
    <p>
      Use this grid to find the session you want
      to attend.  Note that the keynote and lunch
      are in the ballroom.
    </p>
  </section>
```

With that id in place we can alter the <table> tag to reference that descriptive section:

```
<table aria-describedby="schedule_instructions">
```

Including captions and additional descriptions with tables helps people who use screen readers understand the context of the tables more clearly and improves usability for sighted users, as well. The `<caption>` element has been available in browsers for years, and browsers that don't understand the `aria-describedby` attribute will just ignore it, so there's no reason not to use these techniques with data tables right now.

5.1 The Future

HTML5 and the WAI-ARIA specification have paved the way for a much more accessible Web. With the ability to identify changing regions on the page, developers can create richer JavaScript applications without worrying so much about accessibility issues. Thanks to the ease of use, these roles are being included in popular JavaScript frameworks like Ember, jQuery Mobile, and many more, meaning that developers using those frameworks will be automatically building more-accessible applications.