HTML table accessibility

- Recap: Tables for visually impaired users
  - Using column and row headers
- Adding a caption to your table with <caption>
  - Table caption practice
- Adding structure with <thead>, <tbody>, and <tfoot>
  - Adding structure to a spending record table
- The scope attribute
- The id and headers attributes
- Playing with scope and headers
- Summary

In the previous article, we looked at one of the most important features for making HTML tables accessible to visually impaired users — the th element. In this article, we continue down this path, looking at more HTML table accessibility features such as captions/summaries, grouping your rows into table head, body and footer sections, and scoping columns and rows.

Prerequisites:	The basics of HTML (see Basic HTML Syntax).
Learning outcomes:	• An understanding of the accessibility issues associated with tables. • Adding captions to tables. • Better table structuring with head, body, and footer. • Creating further association between headers and cells with the `scope`, `id`, and `headers` attributes.

Recap: Tables for visually impaired users

Let's recap briefly on how we use data tables. A table can be a handy tool, for giving us quick access to data and allowing us to look up different values. For example, it takes only a short glance at the table below to find out how many rings were sold in Ghent during August 2016. To understand its information, we make visual associations between the data in this table and its column and/or row headers.

Items Sold August 2016

		Clothes			Accessories	
		Trousers	Skirts	Dresses	Bracelets	Rings
Belgium	Antwerp	56	22	43	72	23
	Ghent	46	18	50	61	15
	Brussels	51	27	38	69	28
The Netherlands	Amsterdam	89	34	69	85	38
	Utrecht	80	12	43	36	19

But what if you cannot make those visual associations? How then can you read a table like the above? Visually impaired people often use a screen reader that reads out information on web pages to them. This is no problem when you're reading plain text but interpreting a table can be quite a challenge for a blind person. Nevertheless, with the proper markup we can replace visual associations by programmatic ones.

> **NOTE:** There are around 253 Million people living with Visual Impairment according to WHO data in 2017.

Using column and row headers

Screen readers will identify all headers and use them to make programmatic associations between those headers and the cells they relate to. The combination of column and row headers will identify and interpret the data in each cell so that screen reader users can interpret the table similarly to how a sighted user does.

We already covered headers in our previous article — see Adding headers with <th> elements.

Adding a caption to your table with <caption>

You can give your table a caption by putting it inside a caption element and nesting that inside the table element. You should put it just below the opening `<table>` tag.

```
<table>
  <caption>
    Dinosaurs in the Jurassic period
  </caption>
  <!-- … -->
</table>
```

As you can infer from the brief example above, the caption is meant to contain a description of the table contents. This is useful for all readers wishing to get a quick idea of whether the table is useful to them as they scan the page, but particularly for blind users. Rather than have a screen reader read out the contents of many cells just to find out what the table is about, the user can rely on a caption and then decide whether or not to read the table in greater detail.

A caption is placed directly beneath the `<table>` tag.

> **NOTE:** The `summary` attribute can also be used on the `<table>` element to provide a description — this is also read out by screen readers. We'd recommend using the `<caption>` element instead, however, as `summary` is deprecated and can't be read by sighted users (it doesn't appear on the page).

Table caption practice

At this point we'll get you to try out adding a caption to an HTML table, using the school timetable you met in the previous article.

1. Copy the first HTML block in the Grouping columns with `<colgroup>` and `<col>` into an HTML file on your computer, or an online editor such as CodePen or JSBin.
2. Add a suitable caption to the table.
3. Save your code and see what it looks like.

Click here to show the solution

Your finished HTML should look something like this:

```
<table>
  <caption>
    St. Winnifred's weekly language lesson timetable
  </caption>
  <colgroup>
    <col span="2" />
    <col class="column-background" />
    <col class="column-fixed-width" />
    <col class="column-background" />
    <col class="column-background-border" />
    <col span="2" class="column-fixed-width" />
  </colgroup>

  <!-- Rest of code omitted for brevity -->
</table>
```

Adding structure with <thead>, <tbody>, and <tfoot>

As your tables get a bit more complex in structure, it is useful to give them more structural definition. One clear way to do this is by using thead, tbody, and tfoot, which allow you to mark up a header, body, and footer section for the table.

These elements don't necessarily make the table any more accessible to screen reader users. They don't result in any visual enhancement on their own, however they are very useful for applying styling and layout enhancements via CSS, which can improve accessibility. To give you some interesting examples, in the case of a long table you could make the table header and footer repeat on every printed page, and you could make the table body display on a single page and have the contents available by scrolling up and down.

To use them, they should be included in the following order:

- The `<thead>` element must wrap the part of the table that is the header — this is usually the first row containing the column headings, but this is not necessarily always the case. If you are using col/colgroup elements, the table header should come just below those.
- The `<tbody>` element needs to wrap the main part of the table content that isn't the table header or footer, and should come after the `<thead>`.

- The `<tfoot>` element needs to wrap the part of the table that is the footer — this might be a final row with items in the previous rows summed, for example. The `<tfoot>` should come after the `<tbody>`.

> **NOTE:** `<tbody>` is always implicitly included in every table if you don't specify it in your code. To check this, open up one of your previous examples that doesn't include `<tbody>` and look at the HTML code in your browser developer tools — you will see that the browser has added this tag for you. You might wonder why you ought to bother including it at all — you should, because it gives you more control over your table structure and styling.

Adding structure to a spending record table

Let's get you to put these new elements into action.

1. First of all, create a new HTML file called `spending-record.html` and put the following HTML inside the `<body>`:

```
<h1>My spending record</h1>

<table>
  <caption>
    How I chose to spend my money
  </caption>
  <tr>
    <th>Purchase</th>
    <th>Location</th>
    <th>Date</th>
    <th>Evaluation</th>
    <th>Cost (€)</th>
  </tr>
  <tr>
    <td>Haircut</td>
    <td>Hairdresser</td>
    <td>12/09</td>
    <td>Great idea</td>
    <td>30</td>
```

```
  </tr>
  <tr>
    <td>Lasagna</td>
    <td>Restaurant</td>
    <td>12/09</td>
    <td>Regrets</td>
    <td>18</td>
  </tr>
  <tr>
    <td>Shoes</td>
    <td>Shoe shop</td>
    <td>13/09</td>
    <td>Big regrets</td>
    <td>65</td>
  </tr>
  <tr>
    <td>Toothpaste</td>
    <td>Supermarket</td>
    <td>13/09</td>
    <td>Good</td>
    <td>5</td>
  </tr>
  <tr>
    <td>SUM</td>
    <td>118</td>
  </tr>
</table>
```

2. Next, create a CSS file called `minimal-table.css` in the same directory as your HTML file and fill it with the following content:

```
html {
  font-family: sans-serif;
}

table {
  border-collapse: collapse;
  border: 2px solid rgb(200 200 200);
  letter-spacing: 1px;
  font-size: 0.8rem;
```

```
}

td,
th {
  border: 1px solid rgb(190 190 190);
  padding: 10px 20px;
}

th {
  background-color: rgb(235 235 235);
}

td {
  text-align: center;
}

tr:nth-child(even) td {
  background-color: rgb(250 250 250);
}

tr:nth-child(odd) td {
  background-color: rgb(245 245 245);
}

caption {
  padding: 10px;
}
```

3. Add a `<link>` element into your HTML `<head>` to apply the CSS to the HTML (see Applying CSS and JavaScript to HTML for help with this).
4. Try putting the obvious headers row inside a `<thead>` element, the "SUM" row inside a `<tfoot>` element, and the rest of the content inside a `<tbody>` element.
5. Next, add a `colspan` attribute to make the "SUM" cell span across the first four columns, so the actual number appears at the bottom of the "Cost" column.
6. Let's add some simple extra styling to the table, to give you an idea of how useful these elements are for applying CSS. Add the following to your CSS file:

```
tbody {
  font-size: 95%;
```

```
  font-style: italic;
}

tfoot {
  font-weight: bold;
}
```

> **NOTE:** We don't expect you to fully understand the CSS right now. You'll learn more about this when you go through our CSS modules (starting with CSS Styling basics, which includes an article specifically on styling tables).

7. Save and refresh, and have a look at the result. If the `<tbody>` and `<tfoot>` elements weren't in place, you'd have to write much more complicated selectors/rules to apply the same styling.

The finished example should look like this:

Click here to show the solution

Your finished HTML should look something like this:

```
<table>
  <caption>
    How I chose to spend my money
  </caption>
  <thead>
    <tr>
      <th>Purchase</th>
      <th>Location</th>
      <th>Date</th>
      <th>Evaluation</th>
      <th>Cost (€)</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Haircut</td>
```

```
      <td>Hairdresser</td>
      <td>12/09</td>
      <td>Great idea</td>
      <td>30</td>
    </tr>
    <tr>
      <td>Lasagna</td>
      <td>Restaurant</td>
      <td>12/09</td>
      <td>Regrets</td>
      <td>18</td>
    </tr>
    <tr>
      <td>Shoes</td>
      <td>Shoe shop</td>
      <td>13/09</td>
      <td>Big regrets</td>
      <td>65</td>
    </tr>
    <tr>
      <td>Toothpaste</td>
      <td>Supermarket</td>
      <td>13/09</td>
      <td>Good</td>
      <td>5</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td colspan="4">SUM</td>
      <td>118</td>
    </tr>
  </tfoot>
</table>
```

The `scope` attribute

The `scope` attribute can be added to the `<th>` element to tell screen readers exactly what cells the header is a header for — is it a header for the row it is in, or the column, for example? Looking back to our spending record example from earlier on, you could unambiguously define the column headers as column headers like this:

```
<thead>
  <tr>
    <th scope="col">Purchase</th>
    <th scope="col">Location</th>
    <th scope="col">Date</th>
    <th scope="col">Evaluation</th>
    <th scope="col">Cost (€)</th>
  </tr>
</thead>
```

And each row could have a header defined like this (if we added row headers as well as column headers):

```
<tr>
  <th scope="row">Haircut</th>
  <td>Hairdresser</td>
  <td>12/09</td>
  <td>Great idea</td>
  <td>30</td>
</tr>
```

Screen readers will recognize markup structured like this, and allow their users to read out the entire column or row at once, for example.

`scope` has two more possible values — `colgroup` and `rowgroup`. These are used for headings that sit over the top of multiple columns or rows. If you look back at the "Items Sold August 2016" table at the start of this section of the article, you'll see that the "Clothes" cell sits above the "Trousers", "Skirts", and "Dresses" cells. All of these cells should be marked up as headers (`<th>`), but "Clothes" is a heading that sits over the top and defines the other three subheadings. "Clothes" therefore should get an attribute of `scope="colgroup"`, whereas the others would get an attribute of `scope="col"`:

```
<thead>
  <tr>
    <th colspan="3" scope="colgroup">Clothes</th>
  </tr>
  <tr>
```

```
    <th scope="col">Trousers</th>
    <th scope="col">Skirts</th>
    <th scope="col">Dresses</th>
  </tr>
</thead>
```

The same applies to headers for multiple grouped rows. Take another look at the "Items Sold August 2016" table, this time focusing on the rows with the "Amsterdam" and "Utrecht" headers (`<th>`). You'll notice that the "The Netherlands" header, also marked up as a `<th>` element, spans both rows, being the heading for the other two subheadings. Therefore, `scope="rowgroup"` should be specified on this header cell to help screen readers create the correct associations:

```
<tr>
  <th rowspan="2" scope="rowgroup">The Netherlands</th>
  <th scope="row">Amsterdam</th>
  <td>89</td>
  <td>34</td>
  <td>69</td>
</tr>
<tr>
  <th scope="row">Utrecht</th>
  <td>80</td>
  <td>12</td>
  <td>43</td>
</tr>
```

The `id` and `headers` attributes

An alternative to using the `scope` attribute is to use `id` and `headers` attributes to create associations between data cells and header cells.

A `<th>` element can provide a heading for either a data cell (`<td>`) or, in more complex tables, for another header cell (`<th>`). This allows you to create layered or grouped headers, where one header describes several others.

The `headers` attribute is used to link a cell, `<td>` or `<th>`, to one or more header cells. It takes a space-separated list of strings; the order of strings does not matter. Each string must match the unique `id` of a `<th>` element that the cell is associated with.

This method gives your HTML table a more explicit definition of the position of each cell, based on the headers for the column and the row it belongs to, kind of like a spreadsheet. For this to work

well, your table should include both column and row headers.

Let's look at a portion of the "Items Sold August 2016" example to see how to use the `id` and `headers` attributes:

1. Add a unique `id` to each `<th>` element in the table.
2. For the header cells: Add a `headers` attribute to each `<th>` element that acts as a subheading, that is, a header cell with another header above it. The value is the `id` of the high-level heading. In our example, that's `"clothes"` for the column headers and `"belgium"` for the row header.
3. For the data cells: Add a `headers` attribute to each `<td>` element, and add the `id`s of the associated `<th>` element(s) as a space-separated list. You can proceed as you would in a spreadsheet: Find the data cell, then locate the row and column headers that describe it. The order of the specified `id`s doesn't matter, but keeping it consistent helps to keep it organized and improves readability of the code.

```
<thead>
  <tr>
    <th></th>
    <th></th>
    <th id="clothes" colspan="3">Clothes</th>
  </tr>
  <tr>
    <th></th>
    <th></th>
    <th id="trousers" headers="clothes">Trousers</th>
    <th id="skirts" headers="clothes">Skirts</th>
    <th id="dresses" headers="clothes">Dresses</th>
  </tr>
</thead>
<tbody>
  <tr>
    <th id="belgium" rowspan="2">Belgium</th>
    <th id="antwerp" headers="belgium">Antwerp</th>
    <td headers="belgium antwerp clothes trousers">56</td>
    <td headers="belgium antwerp clothes skirts">22</td>
    <td headers="belgium antwerp clothes dresses">43</td>
  </tr>
  <tr>
    <th id="ghent" headers="belgium">Ghent</th>
    <td headers="belgium ghent clothes trousers">41</td>
    <td headers="belgium ghent clothes skirts">17</td>
```

```
    <td headers="belgium ghent clothes dresses">35</td>
  </tr>
</tbody>
```

In this example:

- The `<th>` for `"Belgium"` uses `rowspan="2"` to span both `"Antwerp"` and `"Ghent"`.
- The city header cells (`"Antwerp"` and `"Ghent"`) use the `headers` attribute to reference `"belgium"` to show they belong to the Belgium group.
- Each `<td>` includes a `headers` attribute for country (`belgium`), city (`antwerp` or `ghent`), group (`clothes`), and the specific clothing item (`trousers`, `skirts`, or `dresses`).

> **NOTE:** This method creates very precise associations between headers and data cells but it uses **a lot** more markup and does not leave any room for errors. The `scope` approach is usually sufficient for most tables.

Playing with scope and headers

For this final exercise, we will get you to try using scope and headers on the sample table we introduced above.

1. First make local copies of items-sold.html and minimal-table.css, in a new directory.
2. Try adding in the appropriate `scope` attributes to make this table more accessible.
3. Making another copy of the starter files in another local directory
4. This time make the table more accessible by creating precise and explicit associations using `id` and `headers` attributes.

Click here to show the solution

The first finished HTML example should look something like this:

```
<table>
  <caption>
    Items Sold August 2016
  </caption>
  <thead>
    <tr>
```

```
      <td colspan="2" rowspan="2"></td>
      <th colspan="3" scope="colgroup">Clothes</th>
      <th colspan="2" scope="colgroup">Accessories</th>
    </tr>
    <tr>
      <th scope="col">Trousers</th>
      <th scope="col">Skirts</th>
      <th scope="col">Dresses</th>
      <th scope="col">Bracelets</th>
      <th scope="col">Rings</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th rowspan="3" scope="rowgroup">Belgium</th>
      <th scope="row">Antwerp</th>
      <td>56</td>
      <td>22</td>
      <td>43</td>
      <td>72</td>
      <td>23</td>
    </tr>
    <tr>
      <th scope="row">Ghent</th>
      <td>46</td>
      <td>18</td>
      <td>50</td>
      <td>61</td>
      <td>15</td>
    </tr>
    <tr>
      <th scope="row">Brussels</th>
      <td>51</td>
      <td>27</td>
      <td>38</td>
      <td>69</td>
      <td>28</td>
    </tr>
```

```
    <tr>
      <th rowspan="2" scope="rowgroup">The Netherlands</th>
      <th scope="row">Amsterdam</th>
      <td>89</td>
      <td>34</td>
      <td>69</td>
      <td>85</td>
      <td>38</td>
    </tr>
    <tr>
      <th scope="row">Utrecht</th>
      <td>80</td>
      <td>12</td>
      <td>43</td>
      <td>36</td>
      <td>19</td>
    </tr>
  </tbody>
</table>
```

While the second one should look like this:

```
<table>
  <caption>
    Items Sold August 2016
  </caption>
  <thead>
    <tr>
      <td colspan="2" rowspan="2"></td>
      <th colspan="3" id="clothes">Clothes</th>
      <th colspan="2" id="accessories">Accessories</th>
    </tr>
    <tr>
      <th id="trousers" headers="clothes">Trousers</th>
      <th id="skirts" headers="clothes">Skirts</th>
      <th id="dresses" headers="clothes">Dresses</th>
      <th id="bracelets" headers="accessories">Bracelets</th>
      <th id="rings" headers="accessories">Rings</th>
```

```
    </tr>
  </thead>
  <tbody>
    <tr>
      <th rowspan="3" id="belgium">Belgium</th>
      <th id="antwerp" headers="belgium">Antwerp</th>
      <td headers="antwerp belgium clothes trousers">56</td>
      <td headers="antwerp belgium clothes skirts">22</td>
      <td headers="antwerp belgium clothes dresses">43</td>
      <td headers="antwerp belgium accessories bracelets">72</td>
      <td headers="antwerp belgium accessories rings">23</td>
    </tr>
    <tr>
      <th id="ghent" headers="belgium">Ghent</th>
      <td headers="ghent belgium clothes trousers">46</td>
      <td headers="ghent belgium clothes skirts">18</td>
      <td headers="ghent belgium clothes dresses">50</td>
      <td headers="ghent belgium accessories bracelets">61</td>
      <td headers="ghent belgium accessories rings">15</td>
    </tr>
    <tr>
      <th id="brussels" headers="belgium">Brussels</th>
      <td headers="brussels belgium clothes trousers">51</td>
      <td headers="brussels belgium clothes skirts">27</td>
      <td headers="brussels belgium clothes dresses">38</td>
      <td headers="brussels belgium accessories bracelets">69</td>
      <td headers="brussels belgium accessories rings">28</td>
    </tr>
    <tr>
      <th rowspan="2" id="netherlands">The Netherlands</th>
      <th id="amsterdam" headers="netherlands">Amsterdam</th>
      <td headers="amsterdam netherlands clothes trousers">89</td>
      <td headers="amsterdam netherlands clothes skirts">34</td>
      <td headers="amsterdam netherlands clothes dresses">69</td>
      <td headers="amsterdam netherlands accessories bracelets">85</td>
      <td headers="amsterdam netherlands accessories rings">38</td>
    </tr>
    <tr>
```

```
      <th id="utrecht" headers="netherlands">Utrecht</th>
      <td headers="utrecht netherlands clothes trousers">80</td>
      <td headers="utrecht netherlands clothes skirts">12</td>
      <td headers="utrecht netherlands clothes dresses">43</td>
      <td headers="utrecht netherlands accessories bracelets">36</td>
      <td headers="utrecht netherlands accessories rings">19</td>
    </tr>
  </tbody>
</table>
```

You can also find the finished examples on GitHub:

- For the first example, see items-sold-scope.html (see this running live also).
- For the second example, see items-sold-headers.html (see this running live also).

Summary

There are a few other things you could learn about tables in HTML, but this is all you need to know for now. Next, you can test yourself with our HTML tables challenge. Have fun!

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:32:51 UTC