

# /CSS/produce perfect tables



The table has been discarded by some web designers, but Craig Grannell wants to drag it back into use with strong structural markup, some natty JavaScript and CSS

**Knowledge needed** Intermediate CSS and HTML

**Requires** A text editor

**Project time** 20 minutes

These days, tables are the web element that designers love to hate. “You use tables for layout?” they ask, helpfully following this up with “You idiot!” when a positive response is uttered. However, the truth of the matter when it comes to tables is that, historically, most designers have not only used them for the “wrong” tasks (such as structural page layout), they also haven’t used them to their full potential.

In this tutorial, you’ll find out why the lost art of tables is worth revisiting. I’ll show you how to create a table that includes important accessibility elements, and uses CSS to improve its appearance and boost usability. The table itself is going to be a straightforward four-column design that lists a selection of magazines from Future Publishing, along with their category, editor and current issue at the time of writing.

## Set up the table

Before you get started, copy across the ‘perfect-tables-start-here’ folder from this issue’s CD and open ‘perfect-tables.html’. As you can see, the basics of the page are already taken care of, and it has links to the style sheet and JavaScript document from the ‘perfect-tables-start-here’ folder.

Begin by adding a table element to the page body, including a summary attribute that provides an overview of the table’s content for screen readers. Next, add a caption element directly after the table start tag. The caption element provides a means of associating the table’s title with the table itself, which is

particularly handy for when tables are placed out of context. In such cases, the content may be rendered relatively meaningless, especially for screen reader users.

Once this is done, add three row-group elements: <thead></thead>, <tfoot></tfoot> and <tbody></tbody>, in that order. Getting the order right is important, because the sequence enables a browser to render the table’s footer prior to receiving all of the data. Note that row-groups are always rendered visually in this order: ‘head, tbody, tfoot.

```
<table summary="A list of magazines by Future Publishing, selected by Craig Grannell.">
  <caption>A selection of magazines by Future Publishing</caption>
  <thead>
  </thead>
  <tfoot>
  </tfoot>
  <tbody>
  </tbody>
</table>
```

Next, the table needs to be populated with data. To save you having to type this in yourself, I’ve provided a handy ‘table-stock.txt’ file on the CD. This has all of the data for the table, with each row comprising four tab-separated components. If you’re lucky, your editor might automate the process of turning this data into a basic table, enabling you to then make a few edits accordingly. If not, first place each line within a table row element (<tr></tr>). Next, place each item of the first line within a table header cell (<th></th>), and each element of subsequent rows within a table data cell (<td></td>). The first two lines are shown below:

```
<tr>
  <th>Magazine</th>
  <th>Category</th>
  <th>Editor</th>
  <th>Current issue</th>
</tr>
<tr>
  <td>.net</td>
  <td>Computing and photography</td>
  <td>Dan Oliver</td>
  <td>165</td>
</tr>
```

With this completed, copy the table header row into the content area of the **thead** element created earlier, then copy the remaining rows into the **tbody** element’s content area. For the footer, add a row with a single data cell that provides a credit or source for the data. As the footer only has a single data cell, it needs to span the entire width of the table, which in the other rows is four cells. So, add a **colspan** attribute to the table footer’s data cell, with a value of 4.

```
<tfoot>
<tr>
  <td colspan="4">Selection compiled by Craig Grannell, so blame him.</td>
</tr>
```

A SELECTION OF MAGAZINES BY FUTURE PUBLISHING			
Magazine	Category	Editor	Current issue
.net	Computing and photography	Dan Oliver	165
3D World	Computing and photography	Jim Thacker	93
Computer Arts	Computing and photography	Garrick Webster	138
Computer Music	Music	Ronan Macdonald	114
Digital Home	Consumer electronics	Dean Evans	n/a
Edge	Entertainment/youth	Margaret Robertson	177
Future Music	Music	Oz Owen	189
MacFormat	Computing and photography	Graham Barlow	184
NGamer	Entertainment/youth	Mark Green	12
SFX	Entertainment/youth	Dave Bradley	159
Total Film	Entertainment/youth	Nev Pierce	130

Selection compiled by Craig Grannell, so blame him.

**Structurally sound** Each element of the table stands out from the others. Separator stripes make it quick and easy to read across rows, and the hover stripe enhances this effect

# Expanding tables

Catering for different text sizes is vital

Many designers only test their sites and site elements using default browser settings, but it's always worth at least increasing the text size of a web page to see how your design fares. When it comes to the example in this tutorial, the table headers look different but still work OK, and the background image gradient fades to white at the top and into a solid colour at the bottom – fine for Internet Explorer's Largest setting for Text Size, and for around four text size increases in the likes of Firefox. Elsewhere, the importance of the line-height setting for the th, td rule becomes clear: as text size increases and cell content wraps, leading needs to be tight or the resulting text will look messy.

A selection of magazines by Future Publishing			
Magazine	Category	Editor	Current issue
.net	Computing and photography	Dan Oliver	165
3D World	Computing and photography	Jim Thacker	93
Computer Arts	Computing and photography	Garrick Webster	138
Computer Music	Music	Ronan Macdonald	114
Digital Home	Consumer electronics	Dean Evans	n/a
Edge	Entertainment/youth	Margaret Robertson	177
Future Music	Music	Oz Owen	189
MacFormat	Computing and photography	Graham Barlow	184
NGamer	Entertainment/youth	Mark Green	12
SFX	Entertainment/youth	Dave Bradley	159
Total Film	Entertainment/youth	Nev Pierce	130

Selection compiled by Craig Grannell, so blame him.

**Poor accessibility** Remove the CSS from our web page and the importance of using table header elements becomes clear. Without them, column headers appear as standard table cells

```
</tr>
</tfoot>
```

## Add accessibility attributes

At this point, you could leave things as they are, but it's possible to enhance accessibility further, even with this simple data table. For each of the header cells, add a scope attribute with the value of col, thereby declaring that these cells are headers for the data cells below them. Another aid for screen readers is to abbreviate lengthy headings by use of the abbr attribute. For example, "Current issue" can be abbreviated to "Issue", saving a word each time a data cell is read.

```
<thead>
<tr>
<th scope="col">Magazine</th>
<th scope="col">Category</th>
<th scope="col">Editor</th>
<th scope="col" abbr="Issue">Current issue</th>
</tr>
</thead>
```

Save the web page and open 'perfect-tables.css'. As you can see, a few rules are already defined, dealing with page defaults (removing margins and padding from all elements, then adding 20 pixels of padding to the page body so that the table doesn't hug the browser window edges) and fonts (setting the default font size to 62.5 per cent, thereby enabling font sizes for specific elements to be set using em values a tenth of the target size in pixels). The third section, "3. tables", is blank, and that's where subsequent rules will be placed.

But first, a decision must be made regarding how the table will be styled. As you can see if you test the page at this point, certain defaults are applied to some elements – the caption and table headers are centred, and the headers are also displayed in bold. This means that even without subsequent styling, headers can be differentiated from other cells, which is incredibly handy. For the table's design, we'll add automated separator stripes (by way of a handy piece of JavaScript), an image behind the header cells, and a border around the table – between each column and between each row-group. Because separator stripes will be added later, no borders are required between table body rows.

## Style the table

Add the following three rules to style the table, the header cells and the data cells. The border-collapse setting of collapse ensures that double borders collapse rather than both being shown. Without this, the border setting for the th, td rule would mean that the border between two cells would be two pixels thick. Note the padding settings for th, td, too – these are quite generous, but because tables can be visually overwhelming, it makes sense to add a little extra padding rather than too little. Finally, the tbody td rule removes top and bottom borders from cells within the tbody section of the table, because they will be made visually distinct using separator stripes.

```
table {
border-collapse: collapse;
}
```

**A SELECTION OF MAGAZINES BY FUTURE PUBLISHING**

Magazine	Category	Editor	Current issue
.net	Computing and photography	Dan Oliver	165
3D World	Computing and photography	Jim Thacker	93
Computer Arts	Computing and photography	Garrick Webster	138
Computer Music	Music	Ronan Macdonald	114
Digital Home	Consumer electronics	Dean Evans	n/a
Edge	Entertainment/youth	Margaret Robertson	177
Future Music	Music	Oz Owen	189

**Zooming in** This is our table when the text size is expanded in a browser. As a rule, it's good to expand the size at least four times, checking for usability changes each time

```
th, td {
border: 1px solid #bbbbbb;
font-size: 1.2em;
padding: 4px 10px;
line-height: 1.2;
}
tbody td {
border-top: 0;
border-bottom: 0;
}
```

Add the following two rules to style the caption and table header cells. The caption needs to be bold and eye-catching, hence the large upper case text, but to ensure that it doesn't distract from the table content, its colour has been knocked back from black to dark grey. Caption spacing has been dealt with using padding, because browsers don't tend to be terribly consistent when it comes to rendering caption margins. For the th rule, a background colour is augmented with a horizontally tiling image – a soft gradient that makes the header cells distinct and also leads the eye downwards. The centred alignment of the header contents is overridden, making all the cell content line up nicely.

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

## Expert tip Keep things simple

In most areas of web design, planning and simplicity are key, and tables are no exception. Before starting work on a table, figure out what information you're trying to provide and the simplest way of providing it. Reduce the complexity of the table as much as possible, perhaps sketching it on paper beforehand. Try to avoid creating empty cells, and for single-header tables, stick to the orientation shown in the tutorial, as it's the most common convention and therefore what most users will expect.

```
>> font-size: 1.4em;
text-transform: uppercase;
padding: 0 0 5px;
color: #333333;
}
th {
background: #e4e5e4 url(table-heading-background.gif) 0 50% repeat-x;
text-align: left;
}
```

To balance the caption, the content in the footer's cell is aligned centrally. Font and colour settings then make the content less prominent than that in other cells, since the footer is less important.

```
tfoot {
background-color: #e2e2e2;
color: #555555;
}
tfoot td {
font-size: 1.0em;
text-align: center;
}
```

### Add automated separator stripes

While you could add a class to every other table row and style stripes that way, this method is time-consuming, especially if you add a new row and then have to amend every subsequent table row start tag. So, it makes sense to automate the stripes, which is what you'll do next. This functionality is provided in a script by Matthew Pennell ([www.thewatchmakerproject.com](http://www.thewatchmakerproject.com)) in his *Stripe your tables the OO way* article. See the updated table demo at [www.thewatchmakerproject.com/zebra.html](http://www.thewatchmakerproject.com/zebra.html).

With the script linked to your web page, you first need to add an `id` value to your table that matches the value in the `window.onload` function at the bottom of the JavaScript document – in this case, it means adding `id="stripedTable1"` to the table start tag. Next, two styles – shown below – are required. The first defines a background colour for the separator stripes; the second, which is optional, defines a style for any row the mouse cursor hovers over. The settings below make the hover row a darker blue colour than that used for the separator stripes, and also inverts the colour of the text (from black to white).

```
tbody tr.alt td {
background: #d8ddf0;
}
tbody tr.over td, tbody tr:hover td {
background: #4f70dc;
```

### Resources Find out more online



**Web Usability**  
Accessibility for tables is briefly covered in this tutorial, but the subject is too complex to cover in substantial detail. Handily, the Web Usability site has a section called Accessible Data Tables, and it's well worth a visit.  
[www.usability.com.au/resources/tables.cfm](http://www.usability.com.au/resources/tables.cfm)



**W3C Recommendation**  
For a full overview of the relevant elements and attributes when dealing with tables, check out the Tables in HTML Documents section of W3C's website. When using XHTML, simply update the code shown accordingly.  
[www.w3.org/TR/html4/struct/tables.html](http://www.w3.org/TR/html4/struct/tables.html)



**Organised information** The gigography on the [pinkflag.com](http://pinkflag.com) site uses a variation of the technique I've shown you, making it easy to locate data in a large table of concert details

```
color: #ffffff;
}
```

Adding the following rule makes a highlighted row stand out further by adding a drop shadow background image to the top of cells in the following row. Note that, at the time of writing, Safari 2 ignores this rule, and adding an equivalent selector using `:hover` makes bizarre things happen, with shadows "printing" on each row, so I'll avoid that and hope Safari users don't storm the .net office.

```
tbody tr.over+tr td {
background-image: url(shadow.png);
background-repeat: repeat-x;
}
```

The table is now done! Headers are distinct, as is each row, and the hover effect makes it easy to read all of the items from a row. The caption ensures that the table makes sense, even when positioned out of context (and to screen readers). However, the stripe effect only works with a single table, so I'll end with a solution for when you have several tables on the page:

```
window.onload = function() {
ZebraTable.stripe('stripedTable1');
ZebraTable.stripe('stripedTable2');
ZebraTable.stripe('stripedTableN');
}
```

I've duplicated the line in the `window.onload` function in the JavaScript, and changed the values to something unique in each case. Each value can then be used for a single table's `id` attribute on the web page. ●

Headers are distinct, and the hover effect makes it easy to read all of the items from a row



### About the author

Name | Craig Grannell  
Site | [www.snubcommunications.com](http://www.snubcommunications.com)  
Areas of expertise | Information architecture, site concepts, graphics, interface and front-end design  
Clients | Swim~, Rebellion, IDG  
My favourite superhero is ... | Hellboy