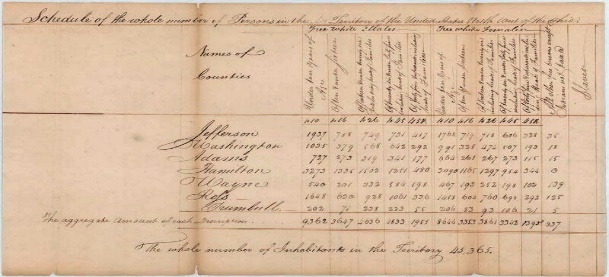
# **HTML table basics**

HTML tables, covering the very basics such as rows and cells, headings, making cells span multiple columns and rows, and how to group together all the cells in a column for styling purposes.

**What is a table?**

A table is a structured set of data made up of rows and columns (tabular data). A table allows you to quickly and easily look up values that indicate connection between different types of data, for example a person and their age, or a day or the week, or the timetable for a local swimming pool.

Tables are very commonly used in human society, and have been for a long time, as evidenced by this US Census document from 1800:

It is therefore no wonder that the creators of HTML provided a means by which to structure and present tabular data on the web.

**How does a table work?**

The point of a table is that it is rigid. Information is easily interpreted by making visual associations between row and column headers. Look at the table below for example and find a Jovian gas giant with 62 moons. You can find the answer by associating the relevant row and column headers.

Data about the planets of our solar system (Planetary facts taken from [Nasa's Planetary Fact Sheet.](https://nssdc.gsfc.nasa.gov/planetary/factsheet/)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Name | Mass  (1024kg) | Diameter (km) | Density (kg/m3) | Gravity (m/s2) | Length of day (hours) | Distance from Sun  (106km) | Mean temper  (°C) |
| Terrestrial planets | | Mercury | 0.330 | 4,879 | 5427 | 3.7 | 4222.6 | 57.9 | 167 |
| Venus | 4.87 | 12,104 | 5243 | 8.9 | 2802.0 | 108.2 | 464 |
| Earth | 5.97 | 12,756 | 5514 | 9.8 | 24.0 | 149.6 | 15 |
| Name | | | | | | | |
|  | | Mars | Mass (10 kg) | Diameter (km) | Density (kg/m) | Gravity (m/s) | Length of day (hours) | Distance from Sun  (10 km) | Mean temper  (°C) |
|  | | Gas  giants | 0.642 | 6,792 | 3933 | 3.7 | 24.7 | 227.9 | -65 |
| Jovian planets | Gas  giants  Ice giants | Jupiter | 1898 | 142,984 | 1326 | 23.1 | 9.9 | 778.6 | -110 |
| Saturn | 568 | 120,536 | 687 | 9.0 | 10.7 | 1433.5 | -140 |
| Ice giants  Pluto | Uranus | 86.8 | 51,118 | 1271 | 8.7 | 17.2 | 2872.5 | -195 |
| Neptune | 102 | 49,528 | 1638 | 11.0 | 16.1 | 4495.1 | -200 |
| Dwarf planets | | Pluto |  |  |  |  |  | 5906.4 | -225 |

When done correctly, even blind people can interpret tabular data in an HTML table — a successful HTML table should enhance the experience of sighted and visually impaired users alike.

**Table Styling:** You can also have a look at a [live example on GitHub!](https://mdn.github.io/learning-area/html/tables/assessment-finished/planets-data.html) One thing you'll notice is that the table does looks a bit more readable there — this is because the table you see above on this page has minimal styling, whereas the GitHub version has more significant CSS applied. Be under no illusion; for tables to be effective on the web, you need to provide some styling information with CSS, as well as good solid structure with HTML. In this excercise we are focusing on the HTML part. In line styling is provided for you to use that will manipulate your tables.

**When should you NOT use HTML tables?** HTML tables should be used for tabular data — this is what they are designed for. Unfortunately, a lot of people used to use HTML tables to lay out web pages, e.g. one row to contain the header, one row to contain the content columns, one row to contain the footer, etc. You can find more details and an example at Page Layouts in our Accessibility Learning Module. This was commonly used because CSS support across browsers used to be terrible; table layouts are much less common nowadays, but you might still see them in some corners of the web. In short, using tables for layout rather than CSS layout techniques is a bad idea. The main reasons are as follows:

1. Layout tables reduce accessibility for visually impaired users: Screen readers, used by blind people, interpret the tags that exist in an HTML page and read out the contents to the user. Because tables are not the right tool for layout, and the markup is more complex than with CSS layout techniques, the screen readers' output will be confusing to their users.
2. Tables produce tag soup: As mentioned above, table layouts generally involve more complex markup structures than proper layout techniques. This can result in the code being harder to write, maintain, and debug.
3. Tables are not automatically responsive: When you use proper layout containers

(such as <header>, <section>, <article>, or <div>), their width defaults to 100% of their parent element. Tables on the other hand are sized according to their content by default, so extra measures are needed to get table layout styling to effectively work across a variety of devices.

## **Creating your first table:** We've talked table theory enough, so, let's dive into a practical example and build up a simple table.

1. Create a new HTML page named Tables. The content of every table is enclosed by these two tags: <table></table>. Add these inside the body of your HTML.
2. The smallest container inside a table is a table cell, which is created by a <td> element ('td' stands for 'table data'). For Example:

<td>Hi, I'm your first cell. </td>

1. The <td> must be wrapped/surrounded by a <tr> tag (tr= table row). For Example:

<tr>

<td>Hi, I'm your first cell. </td>

</tr>

1. **Your Turn:**

<h3>Adding a table to an html document</h3>

<!-- Use the <table> tag to define a table within the <table> tag define as many <tr> tag (table row) and within the <tr> tag define as many <td> tag (table data) each <td> tag defines a column. in general, the first row uses the <th> tag (for column headings) within each <td> tag you write the text that will appear in a table cell-->

<table border="1">

<! --defines the first row-->

<tr>

<! --defines 3 columns (cells) in the first row-->

<td>cell 1-1</td>

<td>cell 1-2</td>

<td>cell 1-3</td>

</tr>

<! --defines the second row-->

<tr>

<! --defines 3 columns (cells) in the second row-->

<td>cell 2-1</td>

<td>cell 2-2</td>

<td>cell 2-3</td>

</tr>

<! --defines the third row-->

<tr>

<! --defines 3 columns (cells) in the third row-->

<td>cell 3-1</td>

<td>cell 3-2</td>

<td>cell 3-3</td>

</tr>

</table>

## **Adding captions with <caption> element:** The <caption> tag defines a table caption. The <caption> tag must be inserted immediately after the <table> tag. *Note: You can specify only one caption per table.* Tip: By default, a table caption will be center-aligned above a table. However, the CSS properties text-align, and caption-side can be used to align and place the caption.

## **Adding headers with <th> elements:** Now let's turn our attention to table headers — special cells that go at the start of a row or column and define the type of data that row, or column contains (as an example, see the "Person" and "Age"). To recognize the table headers as headers, both visually and semantically, you can use the <th> element ('th' stand for 'table header'). This works in the same way as a <td>, except that it denotes a header, not a normal cell.

**Why are headers useful?** We have already partially answered this question — it is easier to find the data you are looking for when the headers clearly stand out, and the design just generally looks better*.* ***Note: Table headings come with some default styling — they are bold and centered even if you don't add your own styling to the table, to help them stand out****.* Tables headers also have an added benefit they allow you to make tables more accessible by associating each header with all the data in the same row or column. Screen readers are then able to read out a whole row or column of data at once, which is useful.

## **Cell Span:** Sometimes we want cells to span multiple rows or columns. Table headers and cells have the colspan and rowspan attributes, which allow us to use as many columns or rows necessary. Both accept a unitless number value, which equals the number of rows or columns you want spanned. For example, colspan="2" makes the cell span two columns.

1. **Now your try:**

## [HTML Character Entities are reserved characters in HTML.](https://www.w3schools.com/html/html_entities.asp)

<hr />

<p><h2>Span of Column & Row</h2></p>

<hr />

<div>

<h3>Col Span</h3>

<p>syntax: &lt;td colspan="number"&gt;here number specifies the number of columns a cell should span.</p>

Note: The placement of the caption tag.

Between the table and the table row

<p> Example: </p>

<table border="1">

<caption>colspan example</caption>

<tr>

<th>header1</th>

<th>header2</th>

<th>header3</th>

<th>header4</th>

</tr>

<tr>

After viewing the table in browser take note of the columns that are offset. To fix the formatting comment lines indicate by <!-- -->

<td colspan="2">data 2,1</td>

<td>data 2,2</td> <!---->

<td>data 2,3</td>

<td>data 2,4</td>

</tr>

<tr>

<td>data 3,1</td>

<td>data 3,2</td>

<td>data 3,3</td>

<td>data 3,4</td>

</tr>

<tr>

<td colspan="4">data 4,1</td>

<td>data 4,2</td> <!---->

<td>data 4,3</td> <!---->

<td>data 4,4</td> <!---->

</tr>

<tr>

<td>data 5,1</td>

<td>data 5,2</td>

<td>data 5,3</td>

<td>data 5,4</td>

</tr>

</table>

</div>

<hr />

<div>

<h3>Row Span</h3>

<p>Syntax: &lt;td rowspan="number"&gt; where <i>number</i> is the number of rows a cell should span</p>

<p> Example: </p>

<table border="1">

<caption>rowspan example</caption>

<tr>

<th>header1</th>

<th>header2</th>

<th>header3</th>

<th>header4</th>

</tr>

<tr>

<td rowspan="2">data 2,1</td>

<td>data 2,2</td>

<td rowspan="4">data 2,3</td>

<td>data 2,4</td>

</tr>

<tr>

<td>data 3,2</td>

<td>data 3,4</td>

</tr>

<tr>

<td>data 4,1</td>

<td>data 4,2</td>

<td>data 4,4</td>

</tr>

<tr>

<td>data 5,1</td>

<td>data 5,2</td>

<td>data 5,4</td>

</tr>

</table>

</div>

<hr />

<div>

<h3>Both rowspan & colspan</h3>

<p> You can use rowspan and colspan at the same time in any table. You can apply rowspan and colspan to any &lt;td&gt; tag to expand a cell both row wise and column wise.</p>

<p>Example: Expand cell data 2,2 to take up six cells (2rows by 3cols)</p>

<table border="1">

<caption>rowspan & colspan example</caption>

<tr>

<th>header1</th>

<th>header2</th>

<th>header3</th>

<th>header4</th>

</tr>

<tr>

<td>data 2,1</td>

<td rowspan="2" colspan="3">data 2,2</td>

</tr>

<tr>

<td>data 3,1</td>

</tr>

<tr>

<td>data 4,1</td>

<td>data 4,2</td>

<td>data 4,3</td>

<td>data 4,4</td>

</tr>

<tr>

<td>data 5,1</td>

<td>data 5,2</td>

<td>data 5,3</td>

<td>data 5,4</td>

</tr>

</table>

</div>

## **Inline Styling:** One more feature to add in for today. HTML has a method of defining styling information for all HTML elements. Take the following example:

<div style=margin: 0 0 20px 20px;> <!--top, right, bottom, left-->

<p> Example: </p>

<table style="width: 100%; max-width: 100%;border: 3px solid blue;padding:5px 5px 5px 5px;">

<caption style="font-size:x-large;font-weight:bolder;">Inline Table Style</caption>

<tr>

<th>Heading 1</th>

<th style="background-color:#213f51;border-color:black;font-size:xx-large;

font-family:Chiller;color:white;text-align:center;">Heading 2</th>

</tr>

<tr>

<td>no style cell</td>

<td style="background-color:#653f79;border-bottom-width: 2px;text-align:right; color:white;">styled cell</td>

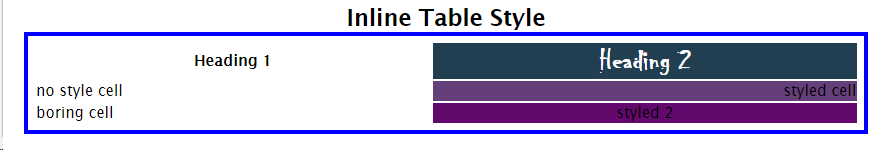
</tr>

<tr>

<td>boring cell</td>

<td style="background-color:#63086d;border-bottom-width:2px;text-align:center; color:white; ">styled 2</td>

</tr>

 </table>

</div>

*Web view of inline styled table:*

This isn't ideal, as we must repeat the styling information across all three cells in the column (we'd probably have a class set on all three in a real project and specify the styling in a separate cascading stylesheet). Instead of doing this, we can specify the information once, on a <table> element, <th> element, along with all the elements with the table.

**What is CSS?**

CSS stands for Cascading Style Sheets.CSS describes how HTML elements are to be displayed. CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

**CSS Solved a Big Problem!**

HTML was NEVER intended to contain tags for formatting a web page. HTML was created to describe the content of a web page. When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process. To solve this problem, the World Wide Web Consortium (W3C) created CSS. CSS removed the style formatting from the HTML page! The style definitions are normally saved in .css files. With a stylesheet file, you can change the look of an entire website by editing one file instead of every page of a website.