HTML Intermediate Guide

htmldog.com

Whereas the purpose of the HTML Beginner's Guide was to teach the **bare essentials**, this guide adds a few nuts and bolts, which shouldn't be particularly difficult as such, but will add a bit more to our understanding of HTML and enable us to do a few more things.

The Intermediate and Advanced Guides are not designed to be followed in any particular order - each page should stand alone.

Span and Div

HTML is all about applying **meaning** to content. Whereas most HTML tags apply meaning (`p` makes a paragraph, `h1` makes a heading etc.), the `span` and `div` tags apply no meaning at all, which might sound about as useful as a foam hammer, but they are actually used quite extensively in conjunction with CSS.

They are used to group a chunk of HTML and hook some information on to that chunk, most commonly with the attributes **class** and **id** to associate the element with a class or id CSS selector.

The difference between `span` and `div` is that a `span` element is **in-line** and usually used for a small chunk of in-line HTML whereas a `div` (division) element is **block-line** (which is basically equivalent to having a line-break before and after it) and used to group larger chunks of code.

```
<div id="scissors">
<p>This is <span class="paper">crazy</span></p>
</div>
```

`div` and especially `span` shouldn't actually be used that often. Whenever there is a sensible alternative that should be used instead. For example, if you want to emphasize the word 'crazy' and the class 'paper' is bold, then the code might look like this:

```
<div id="scissors">
<p>This is <strong class="paper">crazy</strong></p>
</div>
```

If you haven't got a clue about classes and ID's yet, don't worry, they're all explained in the CSS Intermediate Guide. All you need to remember is that `span` and `div` are 'meaningless' tags.

Meta Tags

Once upon a time, many eons ago when the Internet was just a small number of cardboard boxes attached to each other with string, **meta tags** were the town criers of the internet... erm... town.

Meta tags don't do anything to the content that is presented in the browser window, but they are used by the likes of search engines to catalogue information about the web page.

There is one `meta` tag which can be used as many times as you desire and can contain the attributes `content` (required), `http-equiv` and `name`.

The `content` attribute is the meta data itself, which is linked to the name or `http-equiv` attribute.

The name attribute can be anything you like. Commonly used names include `author`, `keywords` and `description`. `description` meta data is often used by search engines (such as Google) to display a description of a web page in its search results, and as such this is perhaps the most useful application of the meta tag.

The `http-equiv` attribute can be used instead of the name attribute and will make an **HTTP header**, which is information sent to the server where the web page is held. The `content` attribute can be `content-type`, `expires`, `refresh` (how often the page automatically refreshes - very bad for accessibility), or `set-cookie`.

```
<html>
<head>
<title>Title</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" /><meta name="description" content="This is my bloody exciting web page about air conditioners" />
...
```

Note

The reason why `meta` tags used to be so important was because they were relied on by search engines to build a profile of a web page. The `keywords` meta data was used extensively for example. Nowadays however, most search engines use the actual content of the page itself, rendering most meta data useless beyond conveying information to someone who is physically reading the HTML.

Tables

Tables may have seemed complicated enough in the HTML Beginner's Guide. It can be quite difficult to visualise a two-dimensional grid from one-dimensional lines of code.

Well, it gets trickier. All thanks to the **rowspan** and **colspan** attributes.

The following code is similar to that in the Tables page of the HTML Beginner's Guide.

```
<table border="1">
    <tr>
        <th>Column 1 heading</th>
        <th>Column 2 heading</th>
        <th>Column 3 heading</th>
    </tr>
    <tr>
        <td>Row 2, cell 1</td>
        <td colspan="2">Row 2, cell 2, also spanning Row 2,
        cell 3</td>
    </tr>
    <tr>
        <td rowspan="2">Row 3, cell 1, also spanning Row 4,
        cell 1</td>
        <td>Row 3, cell 2</td>
        <td>Row 3, cell 3</td>
    </tr>
    <tr>
        <td>Row 4, cell 2</td>
        <td>Row 4, cell 3</td>
    </tr>
</table>
```

Firstly, we have replaced the **td** tags of the first row with **th** tags. Whereas a **td** is a standard **data** cell, **th** is a **header** cell. As with the **td** tag, these tags must be enclosed in **tr** tags.

We have also used **colspan** and **rowspan** attributes in some of the **td** tags. If you look at this code in a browser, you will see that on the second row there are now two cells instead of three, the second cell spanning the second and third column. The **colspan** attribute, which means 'column span' will span the cell over the number of cells that is specified. This means, in this example, there is no need for a third **td** tag - two cells are **merged** into one.

The self-descriptive **rowspan** attribute is similar to **colspan**, except, obviously, it will span across rows rather than columns. Again, the cells that it spans should be removed. In this example, because it spans over the fourth row, there is only two cells in that row.

As with the simpler 3x4, 12-cell table, the numbers on these tables with merged cells should also add up. Although there are only 10 cells in this example, there are 2 spans.

Definition Lists

The HTML Beginner's Guide looked at unordered lists and ordered lists, but much like Peter Cushing's Doctor Who, definition lists are quite often forgotten. This is maybe because they are much more specific than ordered and unordered lists and therefore less useful, but where there is a list of terms and descriptions (such as a glossary), a definition list should be used.

The **dl** element gets the ball rolling, similar to the **ul** and **ol** elements, establishing the list. Rather than there being an **li** element though, definition lists have a **dt** element, which is the **definition term**, followed by a **dd** element which is a **definition description** associated to the **dt** element.

There doesn't have to be one **dt** followed by one **dd**, there can be any number of either. For example, if there are a number of words that have the same meaning, there might be a number of **dt**'s followed by one **dd**. If you have one word that means various different things, there might be one **dt** followed by several **dd**'s.

```html
<h1>Some random glossary thing</h1>
<dl>
    <dt>HTML</dt>
    <dd>Abbreviation for HyperText Markup Language – a language
    used to make web pages.</dd>
    <dt>Dog</dt>
    <dd>Any carnivorous animal belonging to the family
    Canidae.</dd>
    <dd>The domesticated sub-species of the family Canidae,
    Canis lupus familiaris.</dd>
    <dt>Moo juice</dt>
    <dt>Cat beer</dt>
    <dt>Milk</dt>
    <dd>A white liquid produced by cows and used for human
    consumption.</dd>
</dl>
```

Javascript

Javascript is a **client-side scripting language** that can work in conjunction with HTML and while this is not a Javascript guide, we can look at how HTML can *use* Javascript.

Javascript **events** can be used like attributes in HTML tags. An event is something that happens to that HTML element, such as when it is 'clicked' or when it loses focus.

```
<a href="#top" onclick="alert ('wow. Javascript.')">Click me</a>
```

The events that can be used are:
_ **onblur** (used in form elements and executed when an element loses focus)
_ **onchange** (used in form elements and executed when something is changed)
_ **onclick** (executed when a mouse is clicked on an element)
_ **ondblclick** (executed when a mouse is double-clicked on an element)
_ **onfocus** (used in form elements and executed when an element gains focus)
_ **onkeydown** (executed when a key is pressed down)
_ **onkeypress** (executed when a key is pressed and released)
_ **onkeyup** (executed when a key is released)
_ **onload** (used in the body tag and executed when the page loads)
_ **onmousedown** (executed when the button of a mouse is pressed down)
_ **onmousemove** (executed when the mouse cursor moves on an element)
_ **onmouseout** (executed when the mouse cursor moves away from an element)
_ **onmouseover** (executed when mouse cursor moves over an element)
_ **onmouseup** (executed when the button of a mouse is released)
_ **onreset** (used in form elements and executed when a form is reset)
_ **onselect** (used in form elements and executed when an element is selected)
_ **onsubmit** (used in form elements and executed when a form is submitted)
_ **onunload** (used in the body tag and executed when the user navigates away from the page)

Phew.

Note

Try not to get carried away with Javascript. It's best uses tend to be small additions of functionality. There is a danger of seriously degrading the accessibility of a web page with Javascript, and many things that it can be used for can be replaced with server-side scripting languages such as **PHP** or **ASP**.

Bad Tags

This page looks at some of the HTML tags of fairytale worlds and prehistoric times. Bad, nasty, downright ugly little things that belong to outdated HTML standards, random proprietary nonsense that only half-work in one sub-version of one browser or tags that have simply been superseded by newer tags.

Some have suggested that although the approach of HTML Dog to teach standards-based HTML and CSS without making a song and dance about the standards is perhaps a good one, but by doing so, users (beginners in particular) may come across different approaches and bad practices elsewhere without knowing that there is anything wrong with them. So here's HTML Dog's answer: **A guide to what not to use**.

HTML has attempted to move away from the **presentational** and towards the **meaningful**, leading to a philosophy of separating content and meaning (HTML) from presentation (CSS). This general approach tends to lead to much leaner web pages, because a single set of presentational instructions (in an external CSS file) can be applied to many pages. This also makes the site much more manageable because global changes can be made from a single source.

Some of the 'bad tags' are simply presentational tags (such as small) that could be replaced with something meaningful or simply with CSS. Others may not only be presentational, but unnecessarily bulky (such as the font tag) or hideously detrimental to usability (such as blink).

Tags

These are some of the most common tags you might come across that have better alternatives:

_ **b** could be used to make an element bold. Using **strong** (meaning strong emphasis) instead adds meaning, or to just add boldness, **font-weight: bold** in CSS does the job.

_ **i** could be used to italicise an element. Using **em** (meaning emphasis) instead also adds meaning or **font-style: italic** can be used to just add the presentation.

_ **big** could be used to make big text. Using headings instead (**h1**, **h2** etc, when text genuinely is a heading) adds meaning, or simply using the **font-size** property in CSS gives more control.

_ **small** could be used to make small text. CSS (**font-size**) once more gives more control.

_ **hr** could be used to show a horizontal rule. It is unusual to use **hr** in a CSS designed page anyway; properties such as **border-top** and **border-bottom** or even just plain old images do the job much better.

Those tags mentioned above are all compliant with the latest HTML standards but they don't apply any meaning to content, which all good tags should. They could be more useful but they aren't particularly harmful, and might easily be mistaken for innocent butter-wouldn't-melt-in-their-mouth nuggets of pure goodness when standing next to the following filthy tags.

_ `u` could be used to underline elements. It remains that underlined text is still associated by many with links. This is why this tag died a long time ago - you really don't want to be underlining non-linking text.

_ `center` could be used to centre one element within another. The CSS property `text-align` allows values of not only `center`, but `left`, `right` and `justify` as well.

_ `menu` could be used to create a menu list. It does pretty much what `ul` does, but as an 'unordered list' is more general, `ul` stands tall over menu's corpse.

_ `layer` is similar to a `div` element positioned with CSS. These only work in old versions of Netscape. So not very useful then.

_ `blink` or `marquee`. Just say "NO!" kids. They are supposed to do exactly as they say, but have very limited support and were surely only ever intended to be very, very sick jokes.

_ `font`, which could be used to define the font name, size and colour of an element has gained a deserved reputation of being the notoriously mischievous evil goblin lord of Tagworld. Old sites (even some new ones) have `font` tags splattered all over their pages like a plague of termites. Much of their proliferation has come about from web authoring software, placing `font` tags around every element that the web author applied colour or size to. Whereas a `font` tag needs to be applied to every occurrence of an element (say, every time you use a p element), with CSS you can apply properties to every occurrence of an element with just one single little line of code for your whole web site. Using this method, not only is the **page weight substantially lighter** than an equivalent font-tag infested page, but changes can be made more easily because all you need to do is change **one line** of CSS rather than every instance of a `font` tag. This also increases the likelihood of maintaining a consistent design across your site. `font` tags and the inappropriate use of tables are the two most common causes of unnecessarily bloated pages.

Attributes

So you might think you're using the good tags, but there are a few pesky parasitical attributes lurking about that might turn them sour.

_ `name` could be used to assign a name to an element, which is perfectly acceptable in form elements such as `input`, but elsewhere `name`'s job had been replaced by the id attribute.

_ `text` and `bgcolor` could be used to specify the base text colour and background colour of a page within the opening `body` tag. The CSS `color` and `background-color` properties can do this just as well when applied to the `body` selector.

_ **background** could be used within the **body** tag to specify a background image for a page. CSS manages background images much better with properties such as **background-image**.

_ **link**, **alink**, **vlink** could be used within the **body** tag to specify the colour of links (non-visited, active and visited). CSS baby - **:link**, **:active** and **:visited** all do the job.

_ **align** could be used to align the content of an element (such as **<div align="center">Stuff</div>**), but, like the **center** tag, the CSS **text-align** property is the new boss.

_ **target** could be used to open a link in various states, most commonly in a new window (such as **Help me**). Sounds nice, but you're not doing your site any favours. Users don't expect things (such as new windows) to appear as if by magic and the most commonly used navigational tool is the browsers 'back' button, which won't work if you open a link in a new window. It's invalid and it's inaccessible.

Note

Presentational attributes for tags such as **width** and **height** for images and **cellpadding** and **cellspacing** for tables remain due to the frequency that different values need to be applied different elements. They aren't the perfect solution, but if you have a page with a large number of images or tables, you may have no other sensible choice than to use them.

The most baffling presentational attributes belong to the **textarea** tag where not only are the **cols** and **rows** attributes valid, they are required in the latest HTML standard.

Good tags, bad use

To get in to your house you might be able to get down on your knees and squeeze through that little doggy door **but wait!** There's a big ol' human door emblazoned with a device called a door handle! Wow! Look - the door's, like, just the right size for a human to fit through.

The collection of HTML tags (the good ones) was designed for a specific reason and believe it or not (believe it), when you use them for the right reasons, you'll get better results.

Web pages are much more accessible to users with disabilities when the HTML is semantic, as screen readers will often emphasize a list when it encounters a **ul** tag or a heading when it encounters an **h1** or **h2** tag for example.

The most misused HTML of all is tables. Tables are commonly used for

layout, but they should only be used to display tabular data, as they were always only intended to. The non-table layout method isn't some kind of Zen Buddhist quest for true geek enlightenment, there is a real practical benefit of not only dramatically reduced page weight, but also being easier to maintain or redesign.

Note

Sometimes designers will use some of those tags and attributes mentioned here (particularly tables for layout) to achieve a **transitional** design - one that will support older browsers (in particular Netscape 4) as well as modern ones. Tables will indeed allow better presentational control over the CSS inept Netscape 4, but its users are miniscule in number and decreasing while those to whom table-layout would be a great disadvantage - mobile device users - grow in number. The advantages of table-free design mentioned above far outweigh the disadvantages and should result in pages that, although minimally styled to a minority, retain full functionality in **all** browsers.

Frames

Goldilocks thought it would be a really good idea to help herself to a bowl of porridge but then three large carnivorous mammals showed up and threw her out of a window. Frames are bowls of porridge that belong to bears. They might look nice, but it would be quite perilous to go anywhere near them.

Most web sites do not use frames and in general web users are used to a single document as a page.

But if, for some reason, you want to prevent users adding a specific page to their bookmarks or if you want to prevent them recommending a specific page via email or instant messaging or if you want to add a whole other level of complexity to users with disabilities using screen readers who will need to navigate between frames on top of navigating through a page or if you want to confuse the hell out of search engines, go ahead, use frames.

In general, frames do nothing but **add complexity** and **subtract usability**.

Note

If you follow these rules of thumb, you shouldn't go far wrong:

1) If the tag or attribute name even so much as whispers anything suggesting **presentation**, don't use it. That's CSS's job. And CSS does the job better.

2) Use the tag to do what its **name implies**. Tables are for tabular data. Headings are for headings. Etcetera etcetera.

3) When you've got specific content, use the **appropriate** tags. Lists for lists, headings for headings yada yada yada.