



CSS - overview

The thing with tables.....

As you are probably already using tables to control your layout - you might be wondering why they are such a bad idea as a design mechanism when they are being so widely used and generally allow you to achieve the desired result, even if never quite perfect.

Certain considerations are important - though often dismissed;

- loading times
Tables always add to the byte size of your page - nesting tables within tables can soon add up to a heavy loading page. Web browsers are designed to download tables as a single entity which means no content will be displayed until all elements within the table are cached - causing a considerable delay.
- use of inefficient 'placeholder graphics'
Tables usually include transparent GIF files - allowing you as the designer to achieve any layout you desire - by forcing the table into irregular and complex cell configurations. Even though it seems to give you the freedom in layout - these images further slow performance and their maintenance can easily turn into a nightmare as even minor changes 'break' the entire layout.
- maintainance
Tables with a complex array of tables will cost you a lot more in time, money and patience. If you use tools such as Macromedia Dreamweaver or Adobe GoLive to manage your sites and their designs, you can generally ignore the messiness of the nested tables that make the design possible.
But even these tools are not foolproof and don't necessarily present an easy solution. Often behaving radically with unpredictable results - amending the strange layouts of pages they create can be quite a challenge.
If you're like most designers, and you feel you gain more control and understanding if you hand-code everything, then you'll be familiar with another problem complicating the task. HTML coders generally do not force or support the clean indentation of code making it hard to find any given element easily and without problems. And even if you always make sure to indent your code for easier workflow - a simple 'copy&paste' can throw things off at any time.

... and CSS ?!

In contrast, CSS allow you to keep presentation and content relatively separate and therefore making it a lot easier to work with and apply changes and updates at any time.

Through various combinations of selector usage, you have specific control over the way your set styles are applied to any content element/s on the HTML page. You can write a style rule to control overall paragraph appearance as well as control space between image and text elements. In addition to mere control over the layout of page elements - you can add more enhancing features such as the showing in hiding of listed items.



Style sheet rule

Cascading style sheets consist of a collection of rules - each rule consisting of exactly 2 parts:

- 1 • a selector - defining the HTML element/s to which the rule applies
- 2 • a collection of one or more properties (also referred to as attributes) - describing the appearance of all elements in the document matching the selector.

CSS text:

CSS `p {
 colour:#6633CC;
 font-family: 'Verdana', Geneva, Arial, Helvetica, sans-serif;
}`

mark up text:

XHTML `<p> content text to go in here </p>`

* each property consists 2 values, separated by a colon and followed by a semicolon:

- 1 • property of the item
- 2 • value of the property

CSS text:

CSS `p {
 colour:#6633CC;
 font-family: 'Verdana', Geneva, Arial, Helvetica, sans-serif;
}`

mark up text:

XHTML `<p> content text to go in here </p>`

The selector, `p`, indicates the style rule should be applied to all paragraphs in the document.

The `font-family` property is one of a handful of CSS properties to which you can assign a list of possible values, rather than a single, fixed value. When you use a list, commas must separate its individual members.

In this case, the `font-family` property list tells the browser to use Verdana as the font if found on the user's machine. If not, it directs the browser to use Geneva - which if not found will then direct the browser to use Arial... and so on. If neither of these fonts is available, the browser is told to use the default sans-serif type.

Types of style sheets

Cascading style sheets can be used in 3 different ways:

1 inline

= a style defined within an HTML tag (as shown below)

mark up text:

`<h1 style="color:#6633CC; font-weight: bold;"> content text </h1>`

This use of style sheet however is not used frequently and not considered good working practise. Separating content from presentation is one of the main advantages of CSS - embedding styles directly in HTML tags defeats that purpose and not only adds immensely to the byte size of your document but also makes any future update/s time consuming and difficult.

2 embedded

= a style block placed in the head of the document's HTML

mark up text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>embedded CSS</title>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" />
<style type="text/css">
<!--
h1, h2 {
color:#6633CC;
}
h3 {
color:#826798;
}
-->
</style>
</head>
...
```

Even though this method is easy to deal with - it still is a clumsy application of CSS - weighing down the size of the HTML document.

Notice the HTML comment delimiters (`<!-- -->`) just inside the `<style>` tags. These prevent ancient browsers that do not support CSS from interpreting the style rules as document content and displaying them in the browser window. All CSS capable browsers will ignore the comment delimiters. Even though it's probably safe (or nearly so) to omit these symbols today, as so few ancient browsers are still in use, it does no harm to include them - it is best to do so.

3 external

- = an independent style file (".css"), linked to any HTML document using the `<link>` tag, place in the head of the HTML file



mark up text:

XHTML

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>external CSS</title>
<meta http-equiv="Content-Type"
content="text/html; charset=iso-8859-1" />
<link rel="stylesheet" type="text/css" href="sample.css" />
</head>
...
  
```

This is the most flexible and efficient of the three basic methods you can use to insert styles into a web page. If you define an external style sheet file, you can apply it to as many pages on your site as you want. It is as simple as linking to the style sheet from each page on which you want it used.

Making any update or change to the site is easily done by modifying the shared .css file. If you use embedded or, worse yet, inline styles, you'll have to copy and paste them from one HTML file into all other documents.

This method is the easiest way to ensure the maintainability of your CSS styles. If you define all your site's styles in external files, implementing a site-wide style change is a simple matter of making one edit in a single file. All the pages that use that style sheet will display the new styles immediately, following this one change.

With the other techniques, you have to either remember which styles are defined on which pages, or use search mechanisms to help you deal with the decentralized styling rules.

External style sheets are treated as separate files by the browser. When the browser navigates to a new page, using the same style sheet, the external style sheet does not need to be downloaded again. Pages that use external styles are therefore quicker to load.