

Building a basic web-standards compliant, accessible website

Anyone can build a website, and many people do; often with great success, although the majority of websites 'out there' have accessibility issues and are not standards-compliant. You may think this does not matter, and for some personal websites this could be true. However, for commercial (i.e. business) websites, accessibility is a requirement in the UK and many other countries, and, it pays to have a site that won't 'break' in some browsers. With a little thought it is far from difficult to ensure that your website is attractive and both robust and accessible.

Firstly, ensure that your website is standards compliant. It sounds daunting, but it is not. And, to build it locally (i.e. on your own computer before uploading to the web) you don't need any software other than Notepad on Windows (or TextEdit (in plain text mode) on Mac or Bluefish or something similar on Linux).

Before proceeding any further, we need to consider exactly what a plain static website is. Primarily it comprises a number of documents that can be viewed in a browser. Each document is a web page. In later articles we will look at more complex issues such as automatically generated pages, and different forms of interactivity, but for now we will concentrate on creating that first static page for our one page web site.

Before considering any design issues we need to construct the basic un-styled page. Although websites can be built in a number of languages the basic language of most web pages is some flavour of HTML (HyperText Markup Language). Currently being used are HTML 4.01, and various versions of XHTML (eXtensible HyperText Markup Language), which is actually a variant of XML. This all sounds very confusing, but in reality it isn't as the basic syntax of these languages is the same. For this exercise we will use XHTML 1.0 Transitional.

Before we begin we need to make a folder called mywebsite and create an empty text files, index.txt. within the folder. Also within the folder we need another folder called CSS containing an empty text file, style.css.

The web page comprises three distinct elements; the doctype declaration, the head and the body. Everything that you see in the browser is contained within the body; the purpose of the doctype declaration and the head is to provide vital information to the browser about the language being used etc. to enable it to 'de-code' the body.

The doctype declaration for our web page will look like this

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
```

This tells us that the document is written in XHTML 1.0 Transitional and that it is in English and then it opens the tag html that allows the page to be read.

The head for our web page will look like this:

```
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>My standards compliant website</title>
<meta name="description" content="standards compliant web page built in
XHTML and CSS" />
<meta name="keywords" content="My website, standards compliant, CSS,
XHTML" />
<link href="CSS/style.css" rel="stylesheet" type="text/css" />
<link href="CSS/print.css" rel="stylesheet" type="text/css" media="print" />
</head>
```

The head is contained between the `<head></head>` tags and contains information such as the title of the page, and some metadata describing the page. Well developed metadata is useful to the search engines such as Google and thus helps the page become more 'visible' on the web.

The head also contains links to external style sheets (in this case `style.css` and `print.css` in a separate folder called `CSS`. These stylesheets (one for appearance on screen and one for appearance when printed) contain all the code that determines the design of the site (colour, typography, relative size of different elements etc.)

The head can also be used to link to external javascript files which can then be referred to in the body. CSS files (stylesheets) describe appearance, whilst javascript files describe behaviour, although with well-written CSS the need for javascript is minimised.

The interesting bit is the body. It is here that we structure our site. Everything within the body must be contained within the tags `<body></body>`. Indeed for the plainest of sites no other tag needs to be used. But we want more than that, so we will break our site down into four constituent parts:

- The masthead or header
- The navigation
- The content, and,
- The footer

By far the best way to create these different elements is by the use of the `<div></div>` tag. The body of our page will thus look like this:

```
<body>
<div></div>
<div></div>
<div></div>
<div></div>
</body>
</html>
```

We now have four different elements to work with. (Also note that to ensure we don't forget about it the closing `</html>` tag is added to the end. This closes the `<html>` tag (`<html xmlns="http://www.w3.org/1999/xhtml">`) at the beginning of the head and thus tells the browser that we have reached the end of the page).

We have four separate `<div>`s but there is nothing to distinguish on from the other. This is where the clever bit comes in. We can name the individual `<div>`s. We can give each of them an `id`:

```
<body>
<div id="header"></div>
<div id="navigation" ></div>
<div id="content"></div>
<div id="footer"></div>
</body>
</html>
```

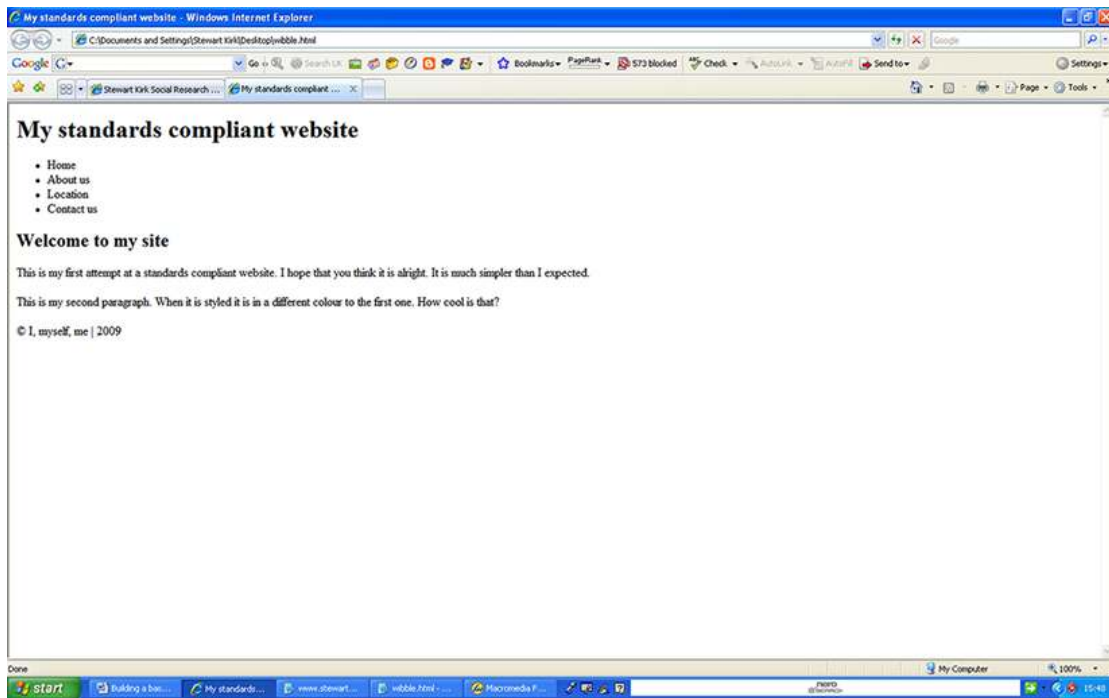
By each having a separate id, when it comes round to writing the CSS we can style each element individually. Now it is time to put some content into our web page. For our content we will want normal paragraphs, headers and some form of navigation. Headers are tags in the form `<h1></h1 >`, `<h2></h2>`, `<h3></h3>` etc and paragraphs are tagged `<p></p>`. For the navigation we will use the unordered list format, which until it is styled looks like this:

- Home
- About us
- Location
- Contact us

This comprises the tag `` for the unordered list and the tags `` for each of the list items.

```
<body>
<div id="header"><h1>My standards compliant website</h1></div>
<div id="navigation" >
<ul>
<li>Home</li>
<li>About us</li>
<li>Location</li>
<li>Contact us</li>
</ul>
</div>
<div id="content">
<h2>Welcome to my site</h2>
<p>This is my first attempt at a standards compliant website. I hope that you think it is alright. It is much simpler than I expected. </p>
<p class="para2">This is my second paragraph. When it is styled it is in a different colour to the first one. How cool is that?</p>
</div>
<div id="footer">&copy; I, myself, me | 2009</div>
</body>
</html>
```

The `©` in the footer is the code for ©, and the second paragraph has been given a class to differentiate it from all other paragraphs. Without styling the page looks a little plain, (see below), but it is a web page divided into elements, capable of being styled.



To sum up thus far, the file index.txt should read as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>My standards compliant website</title>
<meta name="description" content="standards compliant web page built in
XHTML and CSS" />
<meta name="keywords" content="My website, standards compliant, CSS,
XHTML" />
<link href="CSS/style.css" rel="stylesheet" type="text/css" />
<link href="CSS/print.css" rel="stylesheet" type="text/css" media="print" />
</head>
<body>
<div id="header"><h1>My standards compliant website</h1></div>
<div id="navigation" >
<ul>
<li>Home</li>
<li>About us</li>
<li>Location</li>
<li>Contact us</li>
</ul>
</div>
<div id="content">
<h2>Welcome to my site</h2>
<p>This is my first attempt at a standards compliant website. I hope that you
think it is alright. It is much simpler than I expected. </p>
<p class="para2">This is my second paragraph. When it is styled it is in a
different colour to the first one. How cool is that?</p>
</div>
```

```
<div id="footer">&copy; I, myself, me | 2009</div>
</body>
</html>
```

Now save the file as index.html and save it as all files on Notepad (not as a text file in other editors).

We now need to re-open index.html in Notepad and make the following alterations.

```
<ul>
<li>Home</li>
<li>About us</li>
<li>Location</li>
<li>Contact us</li>
</ul>
```

Needs to be changed to read:

```
<ul>
<li><a href="index.html"></a>Home</a></li>
<li><a href="#">About us</a></li>
<li><a href="#">Location</a></li>
<li><a href="#">Contact us</a></li>
</ul>
```

This enables the navigation element to be recognised as navigation (even though thus far we only have one page – we use # as a signifier that there is a link but as yet to nothing).

We now need to turn our attention to the file style.txt

Let us first style the overall background, width and font of the site.

```
/* overall look*/

body {
background-color: #DCDCDC;
width:98%;
font-family: Arial;
font-size: 1em;
text-align:justify;
}
```

We then style the individual elements, header, content and footer.

```
/*header, content and footer*/

#header {
height: 120px;
background-color: #D2928C;
font-family: "Times New Roman";
border: 2px solid red;
text-align:center;
}
```

```
#content{
background-color: #8CABD2;
padding:10px;
}
#footer{
text-align:center;
font-size:0.75em;
margin-top:10px;
width:200px;
margin-left:auto;
margin-right:auto;
border-top: dashed 2px #D2928C;
padding-top:5px;
}
```

And then the typography (the headings and paragraphs).

```
/*typography*/

h1 {
font-size: 2.5em;
padding-top: 0.75em;
color: #865B6A;
}
h2 {
font-size: 1.75em;
color: #D2EAF1;
}
p{
color: #F1D2DA;
}
p.para2{
color: #D2EAF1;
}
```

Note that the colours are written in hexadecimal. They can also be written as RGB and many colours can be referred to by name. These are:

- Aqua
- Black
- Blue
- Fuchsia
- Gray
- Green
- Lime
- Maroon
- Navy
- Olive
- Red
- Purple
- Silver
- Teal
- White
- Yellow

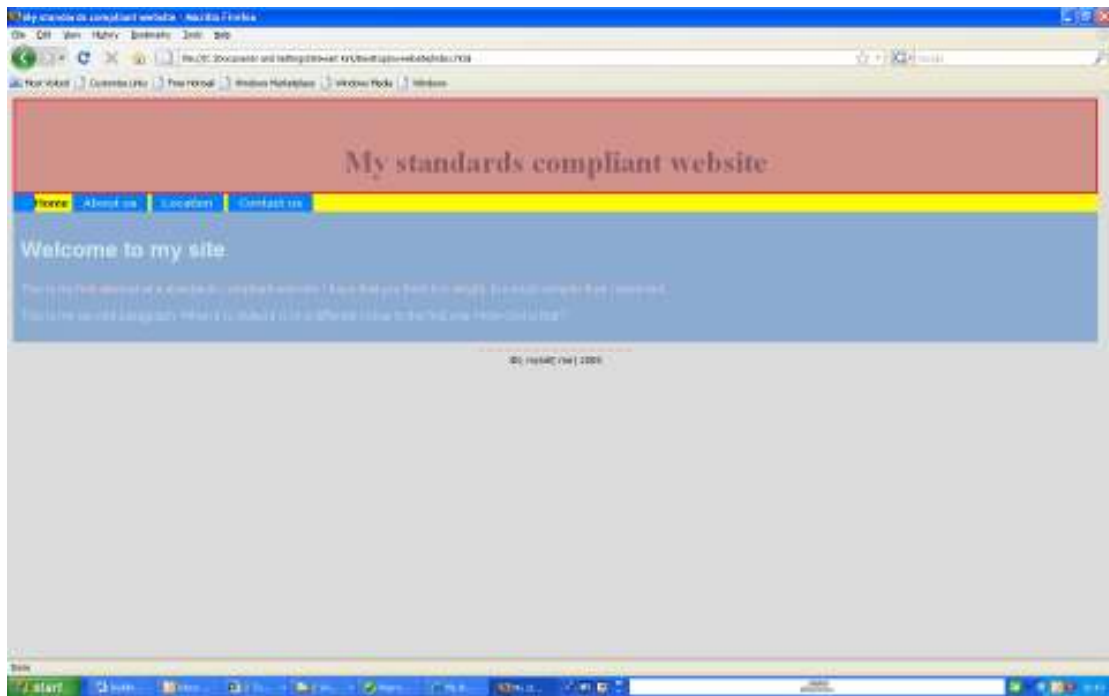
Text-align is self-evident; it is left, right, center, and justify. Padding means the space between the edge of the element and its contents, whilst margin means the gap between the element and adjacent elements.

And finally, the navigation.

```
/*navigation*/

#navigation ul {
padding: 0.2em 0;
margin: 0px 0px 0px 0px;
list-style-type: none;
background: yellow;
font: bold 14px verdana, arial, helvetica, sans-serif;
}
#navigation li {
display : inline;
}
#navigation li a, #navigation a:visited {
text-decoration : none;
background : #0082FF;
color : #f4e1b9;
padding : 0.2em 1em;
}
#navigation a:hover {
text-decoration : none;
background : White;
color : #000;
}
#navigation a:link.active, #tnavigation a:visited.active {
text-decoration : none;
background : Yellow;
color : #000;
}
```

Now save the file as style.css (again saving in Notepad as all files). The web page should now look something like the picture below.



It is not very interesting in terms of design, but styled, albeit in a garish fashion. Try changing elements in the stylesheet to see the changes.

At this stage the page is standards compliant but not necessarily accessible. To ensure accessibility, all images that you may insert into the web page must have an 'alt' tag. This is a short description of the image that enables visually impaired users of the site to be aware of what is there, and enable them to skip the navigation to the and move to the main content. Further all navigation should be navigable without a mouse so you will need to incorporate access keys. The page index.html needs to be amended as follows.

Each of the navigation buttons needs to be given an access key:

```
<ul>
  <li><a href="index.html" accesskey="0"></a>Home</a></li>
  <li><a href="#" accesskey="1">About us</a></li>
  <li><a href="#" accesskey="2">Location</a></li>
  <li><a href="#" accesskey="3">Contact us</a></li>
</ul>
```

Then a small link just after the header to a named anchor at the start of the content needs to be put in place. The header can be altered to read

```
<div id="header"><h1>My standards compliant website</h1>
<p class="access"><a href="index.html#content">skip to main
content</a></p></div>
```

And the content can have an anchor placed at the beginning:

```
<div id="content">
<a name="content" id="content"></a><h2>Welcome to my site</h2>
```

This can be styled out of site using CSS. For example it can be styled off the screen.

```
p.access{
  margin-left:-10000px;
}
```

Index.html will now read:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>My standards compliant website</title>
<meta name="description" content="standards compliant web page built in
XHTML and CSS" /> <meta name="keywords" content="My website,
standards compliant, CSS, XHTML" />
<link href="CSS/style.css" rel="stylesheet" type="text/css" />
<link href="CSS/print.css" rel="stylesheet" type="text/css" media="print" />
</head>
<body>
<div id="header"><h1>My standards compliant website</h1>
<p class="access"><a href="index.html#content">skip to main
content</a></p></div>

<div id="navigation" >
<ul>
<li><a href="index.html" accesskey="0"></a>Home</a></li>
<li><a href="#" accesskey="1">About us</a></li>
<li><a href="#" accesskey="2">Location</a></li>
<li><a href="#" accesskey="3">Contact us</a></li>
</ul>
</div>
<div id="content">
<a name="content" id="content"></a><h2>Welcome to my site</h2>
<p>This is my first attempt at a standards compliant website. I hope that you
think it is alright. It is much simpler than I expected. </p>
<p class="para2">This is my second paragraph. When it is styled it is in a
different colour to the first one. How cool is that?</p>
</div>
<div id="footer">&copy; I, myself, me | 2009</div>
</body>
</html>
```

And style.css will read:

```
/* overall look*/
body {
  background-color: #DCDCDC;
  width:98%;
  font-family: Arial;
  font-size: 1em;
  text-align:justify;
}
```

```

/*header, content and footer*/

#header {
    height: 120px;
    background-color: #D2928C;
    font-family: "Times New Roman";
    border: 2px solid red;
    text-align:center;
}
#content{
    background-color: #8CABD2;
    padding:10px;
}
#footer{
    text-align:center;
    font-size:0.75em;
    margin-top:10px;
    width:200px;
    margin-left:auto;
    margin-right:auto;
    border-top: dashed 2px #D2928C;
    padding-top:5px;
}

/*typography*/

h1 {
    font-size: 2.5em;
    padding-top: 0.75em;
    color: #865B6A;
}
h2 {
    font-size: 1.75em;
    color: #D2EAF1;
}
p{
    color: #F1D2DA;
}
p.para2{
    color: #D2EAF1;
}
p.access{
    margin-left:-10000px;
}
/*navigation*/

#navigation ul {
    padding: 0.2em 0;
    margin: 0px 0px 0px 0px;
    list-style-type: none;
    background: yellow;
    font: bold 14px verdana, arial, helvetica, sans-serif;
}
#navigation li {

```

```
display : inline;
}
#navigation li a, #navigation a:visited {
text-decoration : none;
background : #0082FF;
color : #f4e1b9;
padding : 0.2em 1em;

}
#navigation a:hover {
text-decoration : none;
background : White;
color : #000;
}
#navigation a:link.active, #navigation a:visited.active {
text-decoration : none;
background : Yellow;
color : #000;
}
```

Note that when working with XHTML and CSS the linguistic convention is American English, thus colour is written as color and centre is written as center etc.

At this point the only other thing to look at in terms of basic accessibility is to examine what the site might look like to someone with colour-blindness. There are several excellent sites on the internet that you can access, amongst them being <http://colorfilter.wickline.org/> where you can use a number of different filters to see what your site looks like.

This article has only scratched the surface of XHTML, CSS and web accessibility. There are literally thousands of very useful articles on the web that can be Googled on XHTML / CSS techniques and on improving accessibility. The point of this article is to alert would be website makers that to begin to develop a decent website, there is no need for expensive software, and that to develop a compliant, accessible site is no more difficult than developing a non-compliant, non-accessible site. Now when it comes to making it look good, well you simply need an eye for design.

In later articles I will deal in more detail with the possibilities of CSS for styling, plus looking at elements of interactivity using simple server-side scripting for such things as site search functions and contact forms.

Stewart Kirk, Feb 2009