Semantic code and header tags within DotNetNuke skins

Created by Lee Sykes
DNN Creative Magazine

Version 1.0
Last Updated: 1st July 2006

Sponsors

If you would like to advertise to the DotNetNuke community, we can offer space on www.skinningtoolkit.com. This gives you quality targeted DotNetNuke related advertising at a very low cost. Contact: lee@dnncreative.com for the advertising options and your requirements.

Contents

Semantic code and header tags within DotNetNuke skins

Introduction - What is Semantic code?

HTML was originally intended to describe the contents of a document, not as a way to make your pages appear visually pleasing.

Semantic code returns to this original concept and means that you are using HTML tags for their original purpose rather than styling how the page should look.

For instance, for styling a title you could use:

```
<font size="20px"><strong>This is a title</strong></font>
```

This displays the text in a bold large font so that it appears visually on the page as a title. But, thinking in terms of semantic code there is nothing here that actually describes this piece of text as a title.

Semantically, the code would appear as:

```
<h1>This is a title</h1>
```

So you can easily see within the actual HTML code that this section of text is a title.

To style the title text you would define the h1 tag within your skin.css file.

ie:

```
h1
{
font-size: 20px;
font-weight: bold;
}
```

Benefits of Semantic code

Semantic code makes it easier for various browsers (ie. browsers without style sheets, text browsers, PDAs) and search engines to understand the content of a page.

- Speech browsers for visually impaired people rely on semantic code to understand the content of a page
- Search engines find it easier to index your content, which should improve your search engine results. (A search engine typically will associate a title surrounded by a H1 tag as the main keyword / theme of the page.)

Further benefits include:

- Reduced code required, so pages will load faster, therefore lower bandwidth costs etc
- All styling is completely separate so it’s easy to adjust the style of the entire site by editing the skin.css file rather than on a page by page basis

How do we create semantic code?

The basics:

- For headings / titles use heading tags starting with H1 through to H6
- For paragraphs use p tags
- For lists use list tags

You need to ensure that you use the standard HTML elements that are available rather than style a HTML element to look like another HTML element, for instance not using font tags.

Using header tags

Following the introduction to semantic code, let's look at the use of header tags within the structure of a website page. In particular we are going to apply this to DotNetNuke skin designs.

If you read various CSS web design books, they usually tell you to make use of header tags to structure your web page, but they never offer any suggestions for how to use the header tags (h1, h2, h3, h4, h5, h6). A typical statement would be *'ensure you use header tags starting with h1 through to h6.'*

If you search various forums and articles no-one can offer a definite 'answer.'

Some people consider skipping heading levels to be bad practice. They accept H1 H2 H1 while they do not accept H1 H3 H1 since the heading level H2 is skipped.

Others think that you should not have more than one H1 element on a page.

A book which offers a clue is Wrox's Professional CSS, "*When building your well-meaning markup, it's helpful to think of your document as conforming to a kind of outline – the most important header sits at the top in a h1, beneath it are a number of h2s, beneath each of which is an h3 or two, and so on down to h6. How you envision your document's outline is entirely up to you. Settle upon a model that makes sense to you, and keep it consistent throughout your site's markup"* – page 42.

There are two elements we can pick out here: "*How you envision your document's outline is entirely up to you. Settle upon a model that makes sense to you, and keep it consistent throughout your site's markup"*

This is why there is not a set answer for how to structure your heading tags as every single website is different, but if you keep it consistent for your site it will help users with speech browsers to understand the content of your website.

Analysis of the pros

Following this I have analysed the structure of five websites that are well known for their good design. In particular I have looked at how they have implemented the use of headings to outline the structure of their pages. From this information I will then offer some suggestions for how this can be applied to DotNetNuke skins.

I have analysed two pages from each of these websites, the home page and an article page.

Helpful Tools

If you wish to analyse the document structure and header tags of a website page, the following tools are useful:

Firefox Web Developers Toolbar:
http://chrispederick.com/work/webdeveloper/

Using the Firefox web developers toolbar, go to Information / View Document Outline.

Fangs for Firefox: The Screen Reader Emulator
http://www.standards-schmandards.com/index.php?show/fangs

Fangs creates a textual representation of a web page similar to how the page would be read by a modern screen reader. The extension and source can be downloaded from the SourceForge project page:
http://sourceforge.net/projects/fangs

Once you have installed Fangs to your Firefox browser, right click on the page and select "View Fangs". You will now be able to view your page as if viewed by a screen reader. You can also view the list of headers from the tabs at the top of the page.

Test the Tools

You can view examples of these tools in action by viewing the Document Outline / Fangs outline for these links:

http://www.skinningtoolkit.com/tabid/90/Default.aspx

http://www.skinningtoolkit.com/tabid/112/Default.aspx

If you view the majority of DotNetNuke websites you will notice that they do not make use of heading tags. View www.dotnetnuke.com in Fangs and the Web Developers Toolbar for an example.

Header Tags Structure Analysis

Using the web designer tools we have analysed the following websites:

http://www.alistapart.com

http://www.espn.com

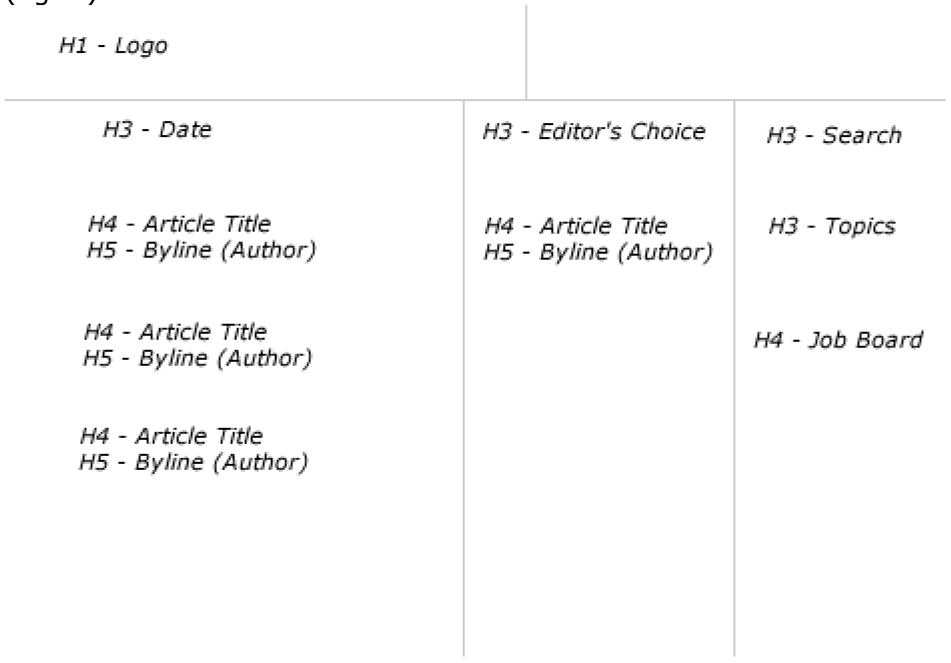
http://veerle.duoh.com/index.php

http://www.thinkvitamin.com/

http://www.webstandards.org/

A List Apart - Home Page:

http://www.alistapart.com/
(fig. 1)



H1 tag surrounds logo of the site

```
<h1 id="masthead">
<a href="/"><img src="/pix/alalogo.gif" alt="A LIST Apart: For People
Who Make Websites" /></a>
</h1>
```

Left column (Main Content)
H3 – Presents today's date (top of main content column)
H4 - Article title - recent articles list, H4 and H5 repeated for each article
H5 – By-line (Article author)

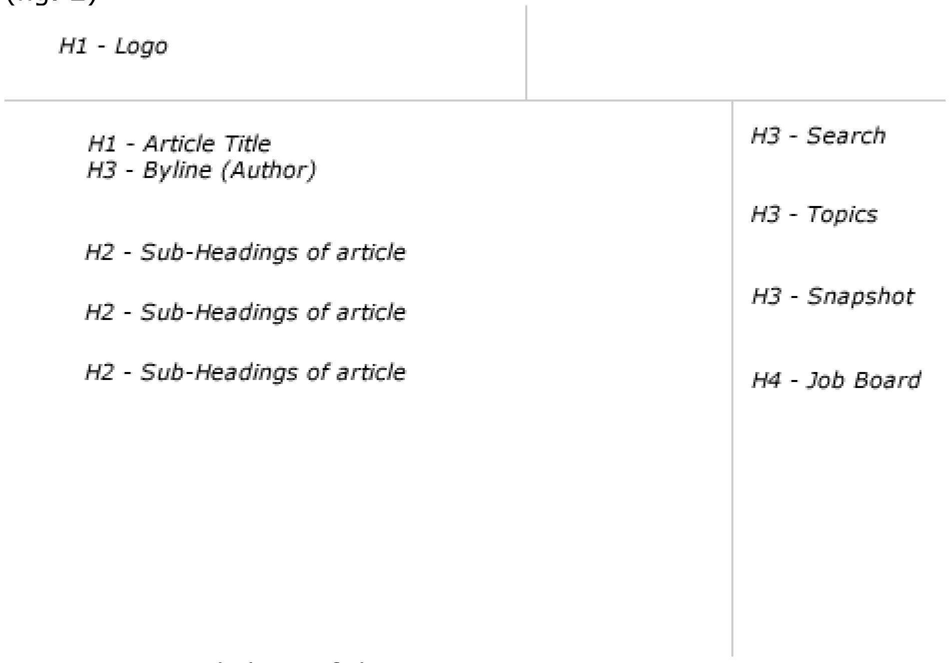
Middle column
H3 – Editor's Choice (Heading for this column of articles)
H4 – Article title
H5 - By-line, Article author

Right column
H3 – Search
H3 – Topics
H4 – Job Board

There are no H2 tags used in the Home Page. This is the only website from the selection that we have analysed that makes use of the H5 tag.

A List Apart - Article Page:

http://www.alistapart.com/articles/worldgrowssmall
(fig. 2)



H1 tag surrounds logo of the site

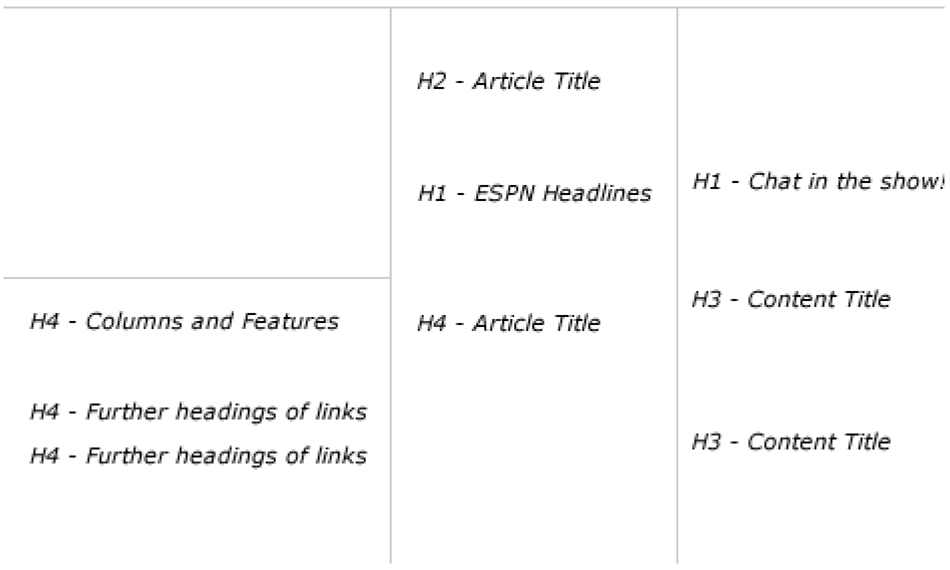
Left column (Main content area)
No header tag around the date
H1 – The Article Title
H3 – By-line (Article author)
H2 – Sub heading of article
H2 – Further sub headings
H3 – Sub-sub headings of article
H2 – Learn More
H2 – Discuss
H2 – About the author

Right Column
H3 – Search
H3 – Topics
H3 - Snapshot
H4 – Job Board

H1 tag is used here for the main title, which is ideal for the search engines as they add further emphasis to any text within H1 tags. H2 and H3 tags are used as sub headings within an article. H3 is used for the title of each container within the right column.

ESPN - Home Page:

http://www.espn.com
(fig. 3)



Right column
H1 – Chat in the Show! (latest chat shows)
H3 – next content header section in right column
H3 – Further heading sections of links in the right column

Left column
H4 – Columns and Features (bottom left column)
H4 – Further heading sections of links in the bottom left column
H4 – More links

Middle column
H2 – Top Middle column item title
H1 – ESPN News Headlines
H4 – Article title

ESPN have used the H1 tag twice. In this case they have used it for the latest headlines and chat shows. It appears as though they are using the Header tags to state the importance of each link section, but the location on the page of these header tags does not seem to match their importance.

For instance, why isn't there a H1 tag, or any header tags in the top left section which is known to be the area that your eye goes to first on the web page? – Based on location importance of a page, you would expect the content in the top left of the page to have the highest priority, yet none of the content uses a header tag.

ESPN have used header tags from H1 to H4 within their home page.

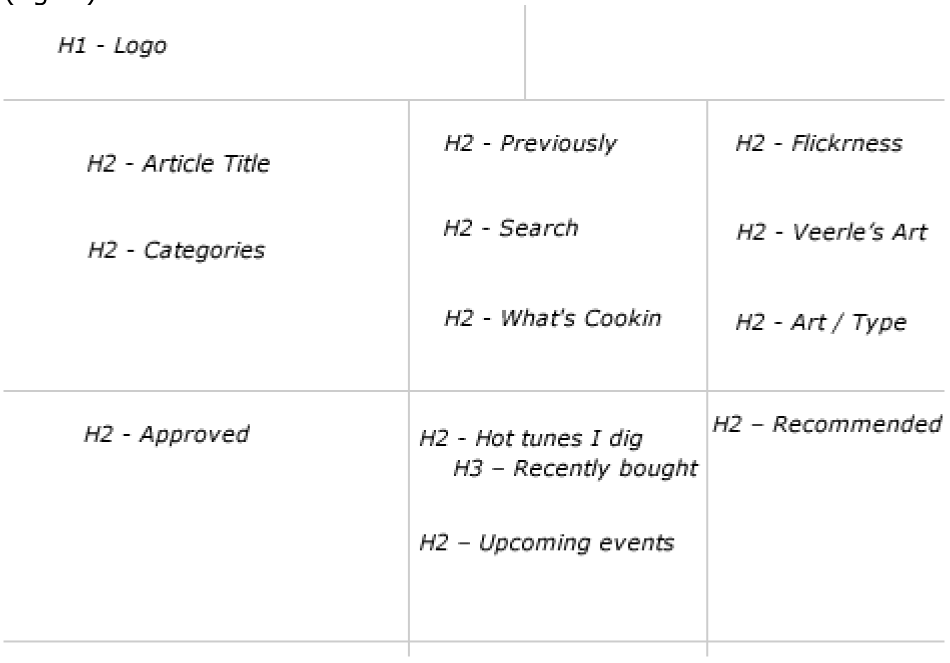
ESPN - Article page:

H1 – Title of article

There are no sub headings used within an article and there are no heading tags used for other sections on the page. There is just one H1 tag used on an article page within ESPN.

Veerle – Home Page:

http://veerle.duoh.com/index.php
(fig. 4)



H1 – Main title of website (graphic header of site)

```
<div id="header"><h1>Veerle's blog</h1></div>
```

Left Column
H2 – Latest post title
H2 – Archive by Category

Middle Column
H2 – Previously
H2 – Search
H2 – What's Cookin here

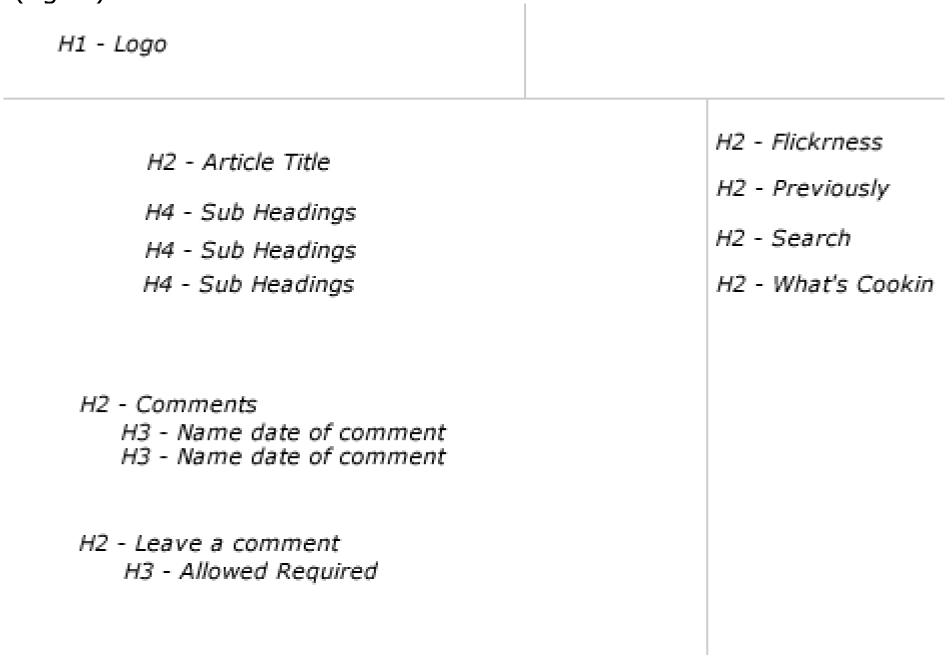
Right Column
H2 – Flickrness
H2 – Veerle's Art
H2 – Art / Type elsewhere

Bottom footer
H2 – Approved
H2 – Hot tunes I dig
H3 – Recently bought at iTMS (sub heading of Hot Tunes)
H2 – Recommended
H2 – Upcoming events

On the home page, the H1 tag is used for the website name and H2 tags are used for headings of each container section.

Veerle - Article Page:

http://veerle.duoh.com/index.php
(fig. 5)



H1 – Main title of website (graphic header of site)

Left column (main content area)
H2 – Article Title
H4 – Sub Heading of article
H4 - Further Sub Headings of article

H2 – Served So Far (Comments section)
H3 – Name date and time of each comment

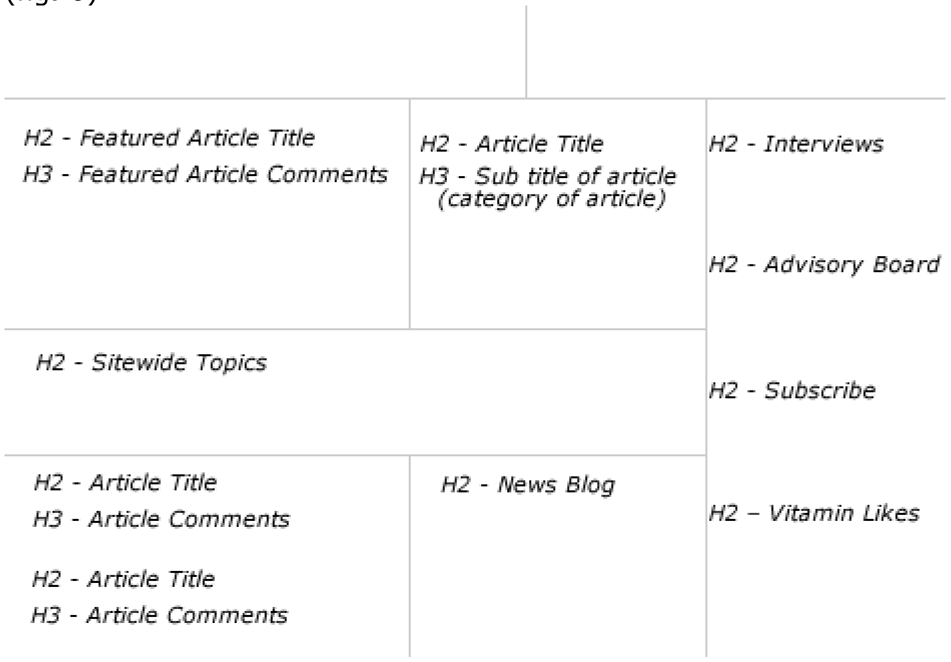
H2 – Leave a comment
H3 – Allowed required (sub heading of leave a comment)

Right Column
H2 – Flickrness
H2 – Previously
H2 – Search
H2 – What's Cookin here

On an article page:
H1 used for title of website
H2 used for article title and for headings of different content sections
H3 used as sub headings for left column under H2 comment sections
H4 used as sub headings for articles

Think Vitamin - Home Page:

http://www.thinkvitamin.com
(fig. 6)



No H1 Tag used

Top left
H2 – Featured article title
H3 – Featured article comments

Top middle
H2 – Article title
H3 – Sub title of article (category of article)

Middle section
H2 – Sitewide topics (listing of categories)

Bottom left column
H2 – Article Title
H3 – Article comments
H2 – Article Title
H3 – Article comments etc.

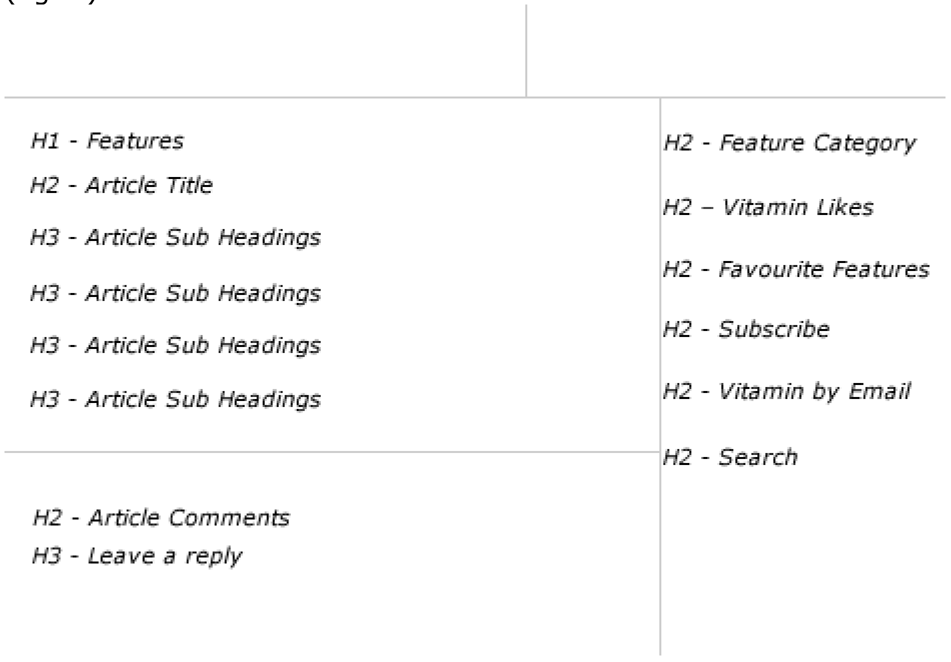
Bottom middle column
H2 – News Blog

Right Column
H2 – Vitamin Interviews
H2 – Advisory Board
H2 – Search etc.

For the Vitamin home page:
H1 – not used
H2 used as article titles for container sections of home page
H3 used as comments link for articles and sub-title (category of an article)

Think Vitamin - Article Page:

http://www.thinkvitamin.com/features/css/stop-css-hacking
(fig. 7)



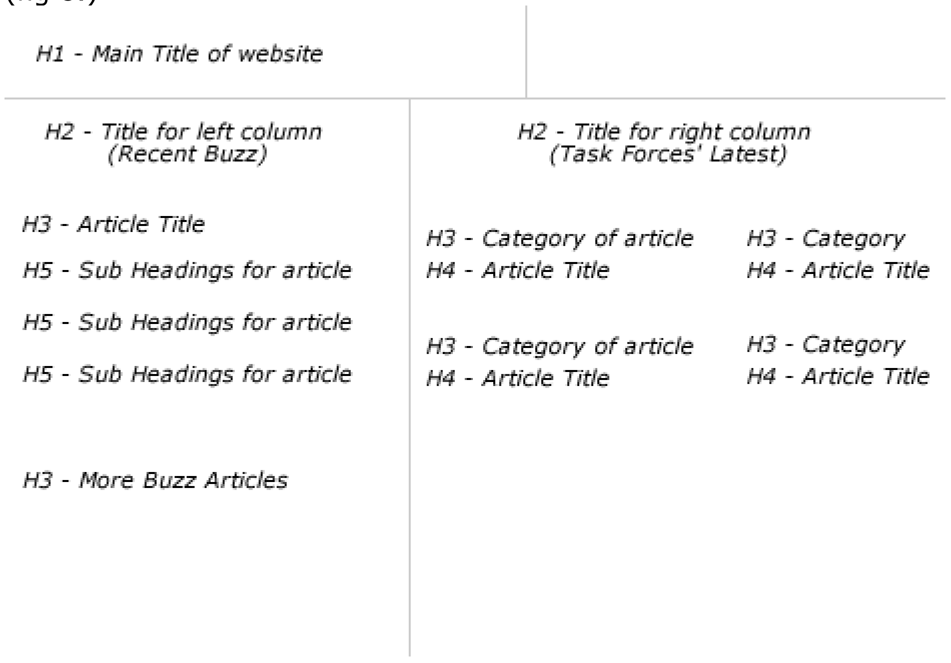
Left Column
H1 – Features
H2 – Article Title
H3 – Sub Headings in the article
H2 – Comments section
H3 – Leave a Reply

Right Column
H2 – Feature categories
H2 – Vitamin Likes
H2 – Favourite Features
H2 – Subscribe
H2 – Vitamin by Email
H2 – Search

For article pages within the Vitamin website:
H1 used to display category of article ie. Features
H2 used for article titles and as titles for different content area sections
H3 sub Headings in an article

Web Standards - Home Page:

http://www.webstandards.org/
(fig 8.)


H1 - Main Title of website
H2 - Title for left column (Recent Buzz)
H3 - Article Title
H5 - Sub Headings for article
H5 - Sub Headings for article
H5 - Sub Headings for article
H3 - More Buzz Articles
H2 - Title for right column (Task Forces' Latest)
H3 - Category of article
H4 - Article Title
H3 - Category
H4 - Article Title
H3 - Category of article
H4 - Article Title
H3 - Category
H4 - Article Title

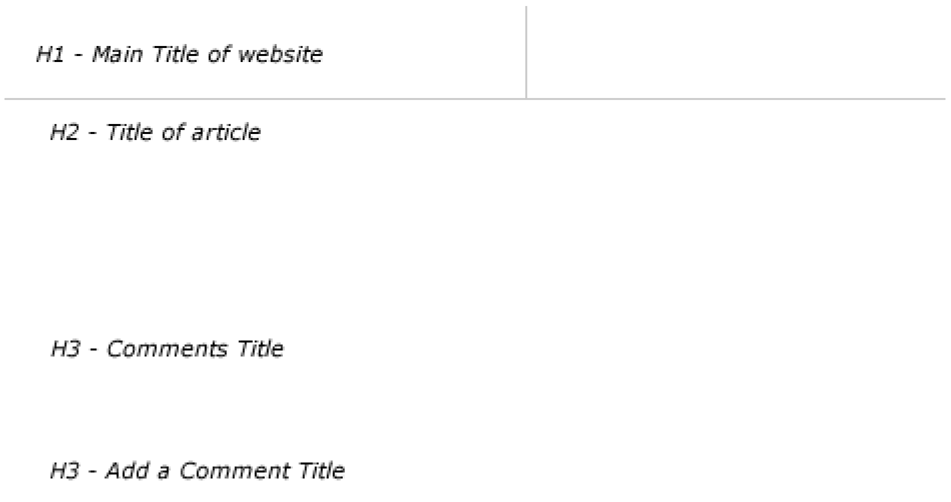
H1 - Main title of website (graphic header of site)

Right Column
H2 – Title for right column (eg. Task Forces' Latest)
H3 – Category of article (eg. Accessibility TF)
H4 – Article title
(H3 and H4 repeated for each category and article)

Left Column
H2 – Title for left column (eg. Recent Buzz)
H3 – Recent article title
H5 – Sub Headings for article
H3 – More buzz articles

Web Standards - Article page:

http://www.webstandards.org/
(fig. 9)



H1 - Main title of website (graphic header of site)
H2 – Title of article
H3 – Comments title
H3 – Add a comment title

DotNetNuke – Header Tags Suggestions and conclusions

Here I am going to offer some suggestions for how you could approach using header tags within your DotNetNuke skins based on the approaches taken in the websites we have analyzed.

Please bear in mind that there is no definite answer for the use of header tags, it all depends on the layout and content within your own website.

From analyzing the various websites we can see that there is not a standard implementation for the use of header tags. We can see however that they have used the header tags to define a 'level of importance' for the various elements on the page and that the header tags are not necessarily listed in tag number order, ie. h1, h2, h3, h4, h5, h6.

We are going to discuss and use this 'level of importance' method within our DotNetNuke skins.

Considerations

H1 tag

Ideally we only want to place one H1 tag on the page. This is so that the search engines can read and understand exactly the topic for the page.

Approaches taken:

From the websites we have analysed we can take two approaches:

1) H1 tag replaced by the logo

Place a H1 tag which contains the main title of the website. This text is replaced by the logo of the website for users with standard browsers, while search engines and screen readers will read the H1 text.

You can view an example of this here: http://veerle.duoh.com/index.php

```
<div id="header"><h1>Veerle's blog</h1></div>
```

Disadvantages: If we hardcode this into the DotNetNuke skin, for every single page the website will have an identical h1 title. Some people state that the h1 tag should display the title of the website and all the following header tags should then display the article titles etc.

I disagree with this approach. The main disadvantage this will bring is related to the search engines. Ideally you would want to create a different h1 title for every single page which will contain a descriptive title with keywords. This can then be aimed directly at search engines helping them to index your page related to your h1 title and therefore increasing your chances of appearing in the search results.

On http://www.alistapart.com/articles/worldgrowssmall

They have used two h1 tags, one around the logo and the other for the article title of an article page. For the home page they use one h1 tag around the logo. I personally would just use one h1 tag to ensure there is no confusion for the search engines.

Advantages: You can set the h1 tag for the entire site and forget about it. If you have a beginner client who is going to be adding their own content to a DotNetNuke portal, you do not have to worry about training them to use h1 tags.

2) H1 tag for the main page title

Place a H1 tag around the main title of the article for the page.

Advantages: The topic will be clear for every single page because every page will have a different keyword related h1 title, meaning improved search engine results.

Disadvantages: With this method, the h1 tag within DotNetNuke will be added by including it in the title of a container. We only want to display one h1 tag per page, so you would need to create a container especially for an h1 tag and another container for all further header tags (We will expand on this later.)

This could potentially be confusing for beginner clients, but if you label each container name appropriately such as "main_article_container" and "sub_container" they should be able to add content in the correct manner without the need for fully understanding the use of header tags.

(Also, as demonstrated later in the tutorial, you can style the H1 container tag text to be different to the other containers title text. This will help to ensure that the H1 tag container is always used as the main article title container.)

Possible Header Tag Structures for the DNN-Blue skin

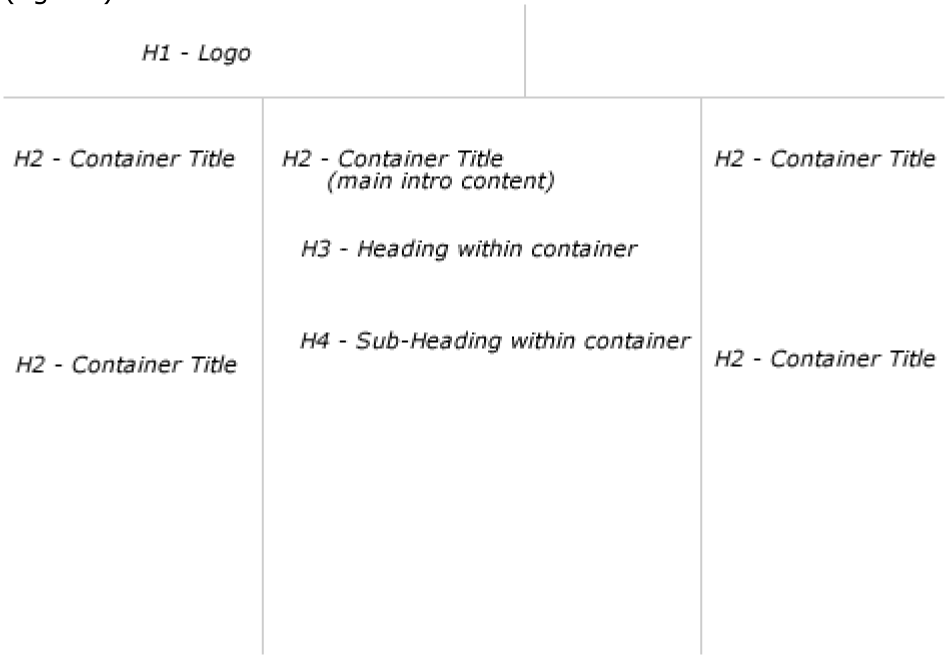
Following the H1 tag possibilities, let's look at some possible header tag structures based on the DNN-blue skin.

Structure One

The first structure that you can see in figure 10 is based on the Home Page of a website using the DNN-Blue skin layout.

The H1 tag is placed around the logo
H2 tags are placed in the title of each container
H3 tags are used as headings within a container
H4 tags are used as sub-headings within a container

DNN-Blue Home Page:
(fig. 10)

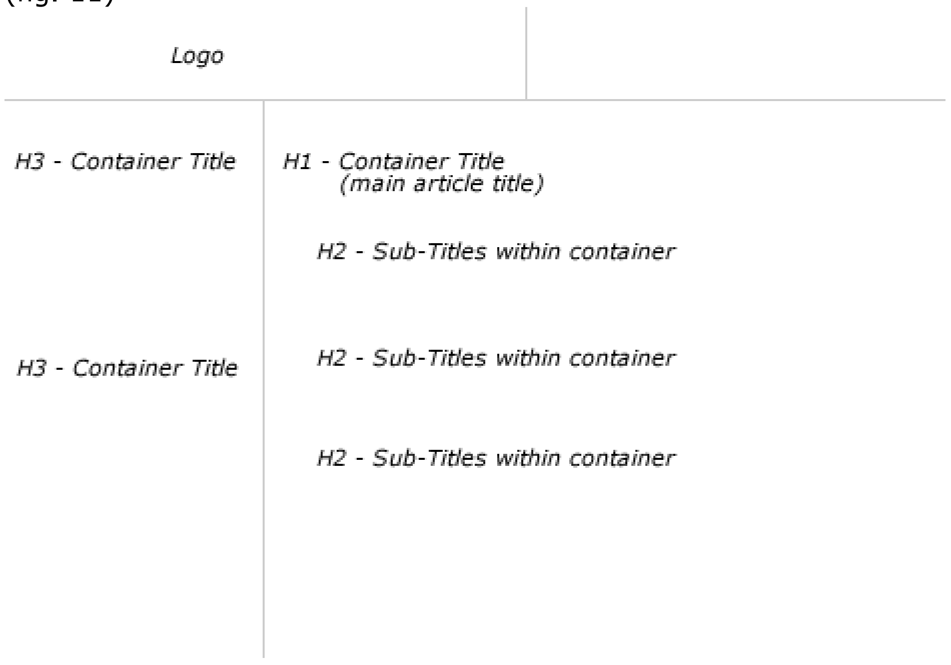


Structure Two

The second structure in figure 11 is based on an Article Page of a website using the DNN-Blue skin.

The logo does not use any header tags
The H1 tag is placed in the title of the container that holds the article
H2 tags are used as sub-headings within the article
H3 tags are placed in the title of any further containers on the page (for example holding an image or links module)

DNN-Blue Article Page:
(fig. 11)


Logo
H3 - Container Title
H1 - Container Title (main article title)
H2 - Sub-Titles within container
H3 - Container Title
H2 - Sub-Titles within container
H2 - Sub-Titles within container

So far we have based these two header layouts on the websites that we have analysed where they used a different header tag structure for a home page and an article page.

For DotNetNuke this brings the disadvantage that we would need to create different skins and containers for the different header tag structures. I personally would want to combine these two header tag layouts into one layout which we can use for the entire website.

This is because I would like to keep the future content management of the site as simple as possible, but, at the same time still flexible enough to manage the layout of different pages.

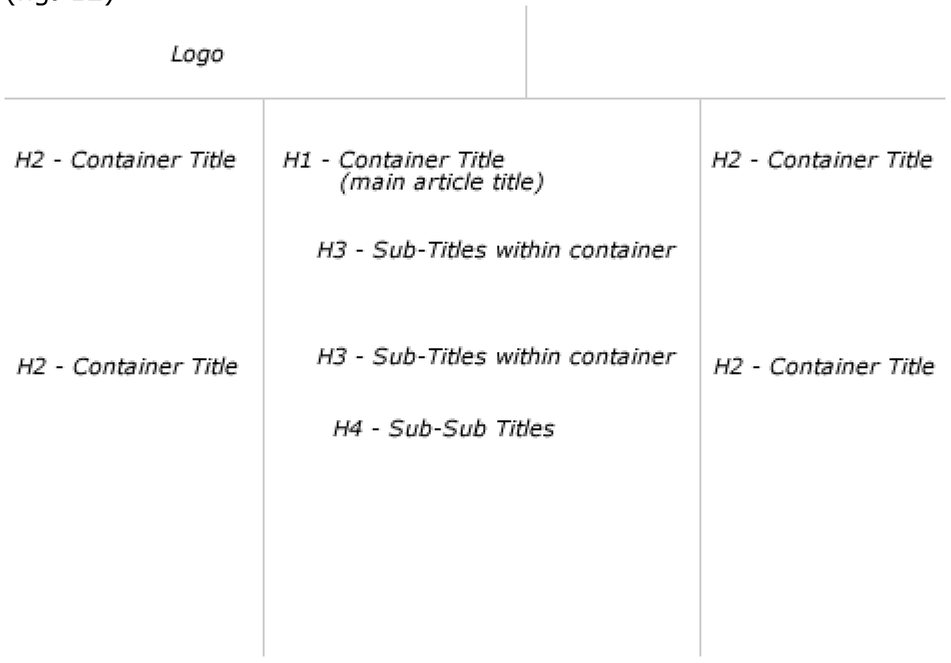
Therefore, the third structure is based on a combination of the Home page and Article page layouts using the DNN-Blue skin.

Structure Three (combined)

The logo does not use any header tags
The H1 tag is placed in the title of the container that holds the article (As discussed earlier, this ensure that every page has a different H1 title, which will improve the search engine results)
H2 tags are placed in the title of any further containers on the page (for example holding an image or links module)
H3 tags are used as sub-headings within the article
H4 tags are used as sub-sub headings within the article

This is a similar structure as used by an article page on http://veerle.duoh.com/index.php

DNN-Blue combined layouts for one skin:
(fig. 12)


Logo
H2 - Container Title
H1 - Container Title (main article title)
H2 - Container Title
H3 - Sub-Titles within container
H2 - Container Title
H3 - Sub-Titles within container
H2 - Container Title
H4 - Sub-Sub Titles

Applying the header tag structure to a DotNetNuke skin

We will now demonstrate how you can apply this header tag structure to the DNN-Blue skin (table based skin).

For this tutorial we will apply the third header tag layout, of course, you can apply any header tag structure that suits your own page layout.

DNN-Blue Skin

Because we are not applying a H1 tag to the logo, we do not need to make any edits to the skin.ascx / skin.htm file.

The changes we need to make are to the Containers of the DNN-Blue skin and the skin.css file.

For this tutorial we will change one of the containers and you can edit any further containers as you need.

DNN-Blue Containers

NOTE: This tutorial assumes you are experienced with creating skins. If you are new to skinning, refer to the skinning tutorials beginning in issue 5 through to issue 9. Starting with How to create a skin for DotNetNuke.

Follow these steps to set up your DNN Containers to use header tags:

Setup the container

1) Create 2 copies of the DNN–Blue image header white background container
2) Name the copies, "image header white background H2" and "image header white background H1"
   (If you are creating these containers for a client, give the containers more meaningful names to the client, ie: "sub_container", "main_article_container")
3) Open "image header white background H2" in your editor of choice. Visual Web Developer Express is a good free tool for editing skins

Edit the container for h2 tags

We are now going to create a container to use the h2 tag. If you refer back to the diagram in figure 12, this container will be used to hold any other content except the main article on the page. ie. content in the left and right panes.

1) Edit line 18 and change:

```
<td valign="middle" width="100%" nowrap>&nbsp;<dnn:TITLE
runat="server" id="dnnTITLE" /></td>
```

to:

```
<td valign="middle" width="100%" nowrap><h2><dnn:TITLE runat="server"
id="dnnTITLE" /></h2></td>
```

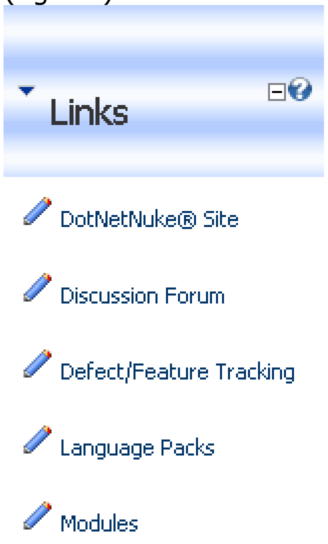
Here we have placed h2 tags around the title of the container.

You will also notice that we have removed the space before the title, "&nbsp;" this causes spacing display problems once we add the h2 tag and is one of the causes for the increased height being displayed around the container title.

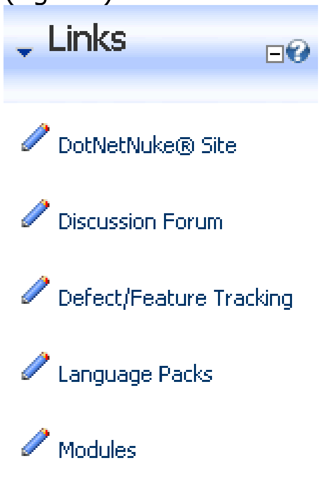
Apart from the &nbsp; causing spacing errors, it is also best practice to remove these spaces and specify any spacing directly within the skin.css file.

View the two example images below:

H2 tag added including the &nbsp;
(fig. 13)



H2 tag added with the &nbsp; removed
(fig. 14)



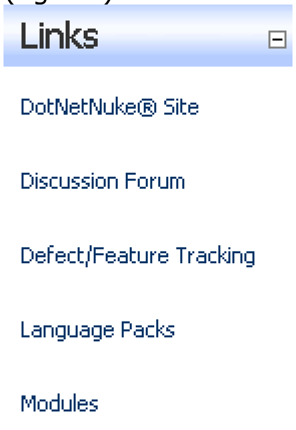
You will see in figure 14 that the header image area still has a height larger than the original. This is caused by the h2 tag. All header tags automatically by default add margins to the top and bottom of the text. We therefore need to remove all of the margins for the h2 tag.

1) Open the skin.css file for the DNN-Blue skin
2) Add the following code:

```
h2
{
    margin: 0;
    padding-left: 3px;
}
```

Here you can see we have removed all margins from the h2 tag and we have also added 3px of padding to the left of the h2 tag. This is to replace the "&nbsp;" that we removed from the skin.ascx file.

The container with margins removed and padding applied
(fig. 15)



We have now completed a container which places h2 tags around the title of the container.

Edit the container for h1 tags

We are now going to create a container to use the h1 tag. If you refer back to the diagram in figure 12, this container will be used to hold the main article content on the page. ie. content in the middle pane.

1) Open "image header white background H1" in your editor of choice.
2) Edit line 18 and change:

```
<td valign="middle" width="100%" nowrap>&nbsp;<dnn:TITLE
runat="server" id="dnnTITLE" /></td>
```

to:

```
<td valign="middle" width="100%" nowrap><h1><dnn:TITLE runat="server"
id="dnnTITLE" /></h1></td>
```

Here we have placed h1 tags around the title of the container.

We have also removed the, "&nbsp;" as we demonstrated for the H2 container.

3) Open the skin.css file for the DNN-Blue skin
4) Change the previous code to:

```
h1, h2
{
    margin: 0;
    padding-left: 3px;
}
```

This groups together the settings for both the h1 and h2 tags.

We have removed all margins from the h1 and h2 tag and we have also added 3px of padding to the left of the h1 and h2 tags.

We have now completed a container which places h1 tags around the title of the container.

Style the H3 and H4 tags

We are going to use the H3 and H4 tags for formatting the titles of our main articles. The main articles will be surrounded by the "image header white background H1" container.

If you look at the default installation content of DotNetNuke you will see the "Administrator Login:" and "Host Login:" sub titles.

We will format both the H3 and H4 tags in this case to look identical to the bold text that is displayed in the default installation content.

1) Open the skin.css file for the DNN-Blue skin
2) Add the following code:

```
h1, h2, h3, h4
{
    margin: 0;
}

h1, h2
{
    padding-left: 3px; /* to replace the &nbsp; in the container file
}

h3, h4 {
    font-family: Tahoma, Arial, Helvetica;
    font-size:  11px;
    font-weight:    bold;
    color: #000;
    margin-bottom: 10px;
}
```

The code above is the complete code so far.

You will see that in the first section of code we have removed all margins for h1, h2, h3, and h4 tags.

We have then moved the padding-left code to a separate section as we do not wish to add any padding to the h3 and h4 tags.

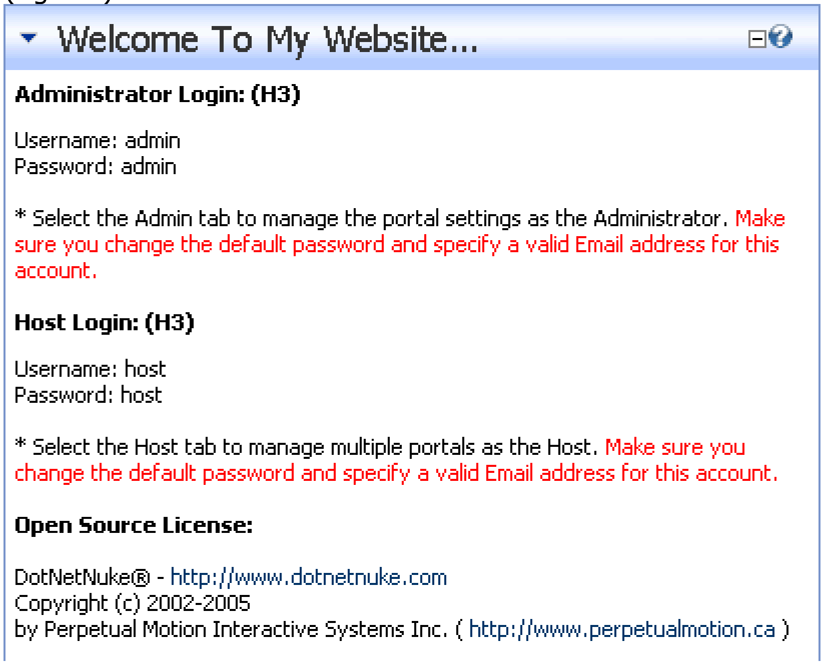
The last section styles the h3 and h4 tags. Here we have specified the full styling for the text as we wish to override the settings within the default.css file. (For further information, view the Introduction to CSS and inheritance in DotNetNuke skins tutorial.

H3 and H4 margins

The first section of code removes all margins from the h3 and h4 tags, in the last section of code we have specified to add a bottom margin of 10 pixels. Overall, this will remove the top margin and then create a bottom margin.

The bottom margin creates the spacing underneath the H3 text. Therefore, we do not need to add a line break into our code (<br />).

H3 tag added
(fig. 16)



Welcome To My Website...

Administrator Login: (H3)

Username: admin
Password: admin

* Select the Admin tab to manage the portal settings as the Administrator. Make sure you change the default password and specify a valid Email address for this account.

Host Login: (H3)

Username: host
Password: host

* Select the Host tab to manage multiple portals as the Host. Make sure you change the default password and specify a valid Email address for this account.

Open Source License:

DotNetNuke® - http://www.dotnetnuke.com
Copyright (c) 2002-2005
by Perpetual Motion Interactive Systems Inc. (http://www.perpetualmotion.ca)

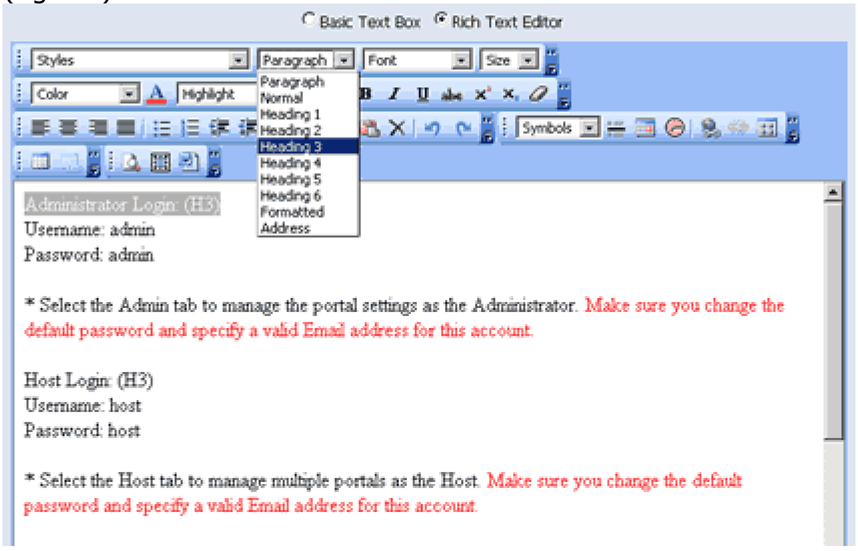
Note that the first two titles use H3 tags, but the final title is styled using bold text. You can see that visually there is no difference between the two.

To format the text within the Text / HTML module

1) Click on Edit text
2) Remove the current default styling that is used for the titles, ie:
3) Click on HTML
4) Remove any bold tags ie. <b> or <strong> from the titles, ie:
   <b>Open Source License:</b>
5) Add the h3 tag to the titles, ie:
6) Click on Design
7) Highlight the text for the title
8) Go to the 'Paragraph' drop down list
9) Select 'Heading 3'
10) Click on update

(View example images below)

Formatting a title to use h3 tags
(fig. 17)



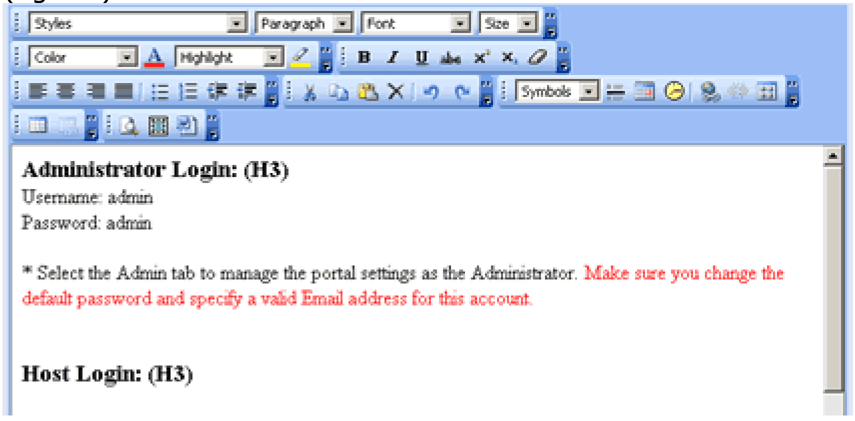
Administrator Login: (H3)
Username: admin
Password: admin

* Select the Admin tab to manage the portal settings as the Administrator. Make sure you change the default password and specify a valid Email address for this account.

Host Login: (H3)
Username: host
Password: host

* Select the Host tab to manage multiple portals as the Host. Make sure you change the default password and specify a valid Email address for this account.

H3 tag placed around titles
(fig. 18)



Administrator Login: (H3)
Username: admin
Password: admin

* Select the Admin tab to manage the portal settings as the Administrator. Make sure you change the default password and specify a valid Email address for this account.

Host Login: (H3)

This creates the view you can see in figure 16.

Further Styling Options

This is where the fun starts!

First of all you need to understand some basic CSS rules.

We can achieve a much finer control for styling the header tags by using the class attribute.

For instance:

(HTML)
```
<h1 class="article">The article title</h1>
```

(CSS)
```
h1.article {
    letter-spacing: 0.2em;
}
```

This will add letter spacing to any h1 titles that use the article class.

Styling the header tags in DotNetNuke

In the Text / HTML module:

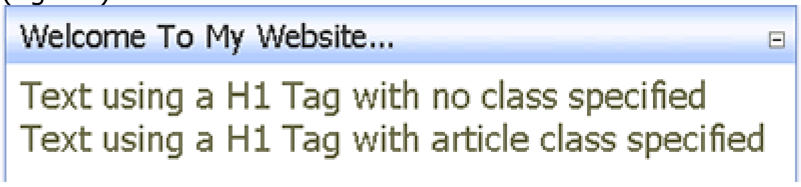
First of all, lets work with some text within the Text / HTML module.

1) Add the following text into the source code of the Text / HTML module

```
<h1>Text using a H1 Tag with no class specified</h1>
<h1 class="article">Text using a H1 Tag with article class
specified</h1>
```

2) Click on Update, your text should be styled as in figure 19.

(fig. 19)



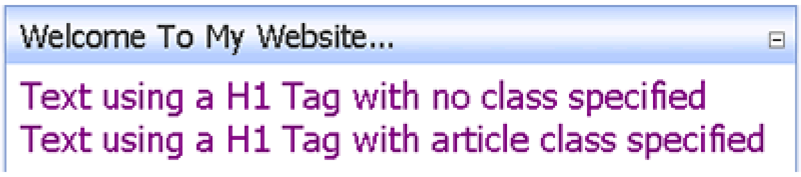
Welcome To My Website...

Text using a H1 Tag with no class specified
Text using a H1 Tag with article class specified

3) Add to the skin.css file

```
h1
{
color: Purple
}
```

This will overide the h1 color styling in the default.css file and all of the text will change to purple.



Welcome To My Website...

Text using a H1 Tag with no class specified
Text using a H1 Tag with article class specified

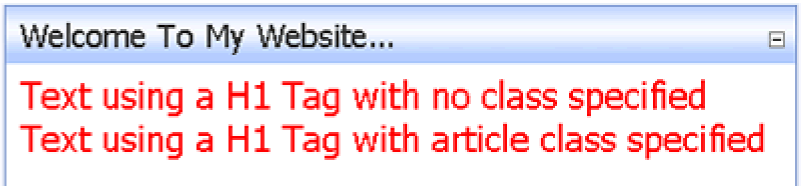
If you view the page source code that DotNetNuke produces, you will see that for the content area within the Text / HTML module, DotNetNuke adds the "Normal" class:

```
<span id="dnn_ctr458_HtmlModule__ctl0_HtmlHolder" class="Normal">
            <h1>Text using a H1 Tag with no class specified</h1>
            <h1 class="article">Text using a H1 Tag with article
class specified</h1>
```

We can take advantage of this and specify styling for any header tags within the Text / HTML module, ie. any h1 tags within the Normal class.

4) Add this code to the skin.css file

```
h1
{
color: Purple
}
.Normal h1
{
color: red;
}
```



Welcome To My Website...

Text using a H1 Tag with no class specified
Text using a H1 Tag with article class specified

You will see that this overrides the styling for a h1 tag and specifies the styling for any h1 tags placed within the Normal class.

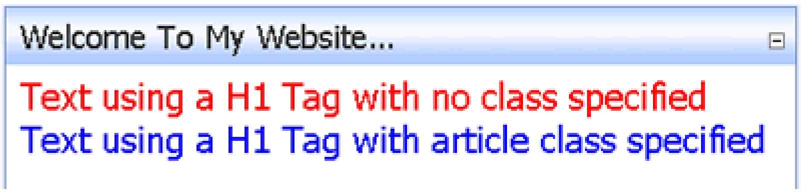
We can take this one step further and specify a class for a header tag.

The second line of text uses the following code:

```
<h1 class="article">Text using a H1 Tag with article class
specified</h1>
```

Add this code to the skin.css file

```
h1
{
color: Purple
}

.Normal h1
{
color: red;
}

h1.article
{
color: blue;
}
```


Welcome To My Website...

Text using a H1 Tag with no class specified
Text using a H1 Tag with article class specified

You will see that any header tags that specify the "article" class are now colored in blue.

Note the order of the CSS code has to be in the order specified above, if you use:

```
h1
{
color: Purple
}

h1.article
{
color: blue;
}

.Normal h1
{
color: red;
}
```

All text will be displayed in red because the .Normal h1 class overrides the h1.article class.

Styling the Module Titles

You will notice that even though we have been making style changes to the h1 tag, this has not changed the styling of the titles for the containers.

If we look at the page source code for the Home page, you will see why:

```
<td valign="middle" width="100%" nowrap>
      <h1><span id="dnn_ctr458_dnnTITLE_lblTitle"
class="Head">Welcome To My Website...</span>
      </h1>
</td>
```

DotNetNuke automatically adds as default a span tag with a "Head" class around each container title, so this overrides any styling that is specified for the h1 or h2 tags that we have placed in our containers.

So, as we have demonstrated in previous tutorials, the .head class within the skin.css file is used as default to style the container title. (View How to create a DotNetNuke Skin and How to Create a DotNetNuke Container tutorials. The DotNetNuke Skinning Toolkit is also a useful reference for learning skin classes.)

We can however specify styling for any h1 tag which has a "Head" class contained within the h1 tags. – Note that this is different to specifying a class for a header tag `<h1 class="head">`

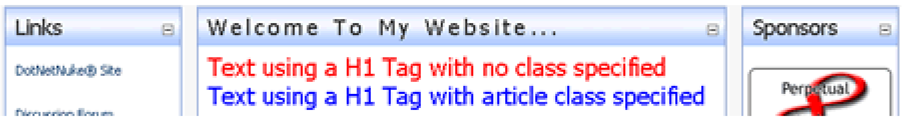
This potentially allows us to set up different styles for different header tags which contain the Head class.

Style the h1 tag for the main title articles

Lets style the text for a main article container title, ie. the "image header white background H1" container title. We will do this by specifying the style for any Head class that is contained within a h1 tag.

Add this code to your skin.css file: (note the space between h1 and .Head)

```
/* head class within the h1 tag */
h1 .Head {
    font-family: Tahoma, Arial, Helvetica;
    font-size:  18px;
    font-weight:   normal;
    letter-spacing: 0.2em;
}
```


Links
DotNetNuke® Site
Discussion Forum
Welcome To My Website...
Text using a H1 Tag with no class specified
Text using a H1 Tag with article class specified
Sponsors
Perpetual

You will see that the "Welcome to My Website" container is using the H1 container and that the text letters are spaced out.

The other containers on the left and right are using the H2 containers and their text is formatted just using the "Head" class, so they do not use spacing.

You will also notice that the h1 tags within the Text / HTML module are not affected by the letter spacing.

Putting it altogether

In the previous example we placed text within the Text / HTML module that used H1 tags. This was to provide an example of the various styling options that are available through manipulating your skin.css file.

To follow the actual header layout that we specified at the beginning of the tutorial (figure 12), we need to do the following:

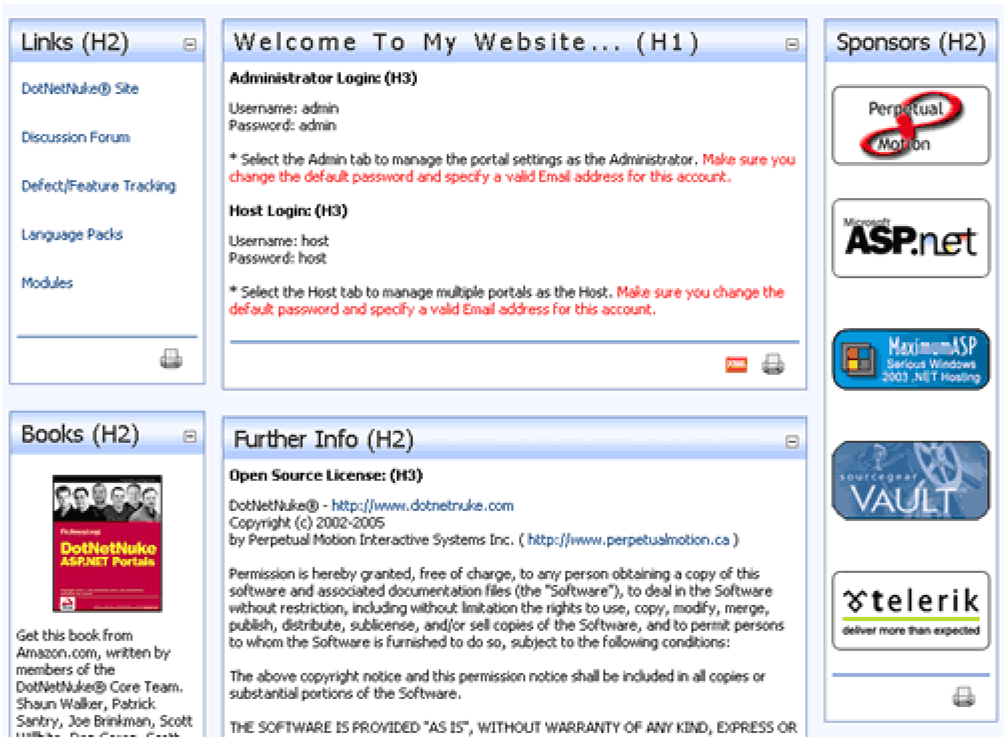
1) In the Left and Right panes use the "image header white background H2" container. – This will place H2 tags around the titles for each of these elements.

2) In the middle content pane, for the top main article use the "image header white background H1" container. This will place H1 tags around the main article title, enabling search engines to easily index the content of the page based on the topic that you specify in the article title.

3) For any further sub titles within the main article, use the H3 and H4 header tags.

4) If you wish to add further containers below the main article container, use the "image header white background H2" container so that this content will have the same importance level as the content in the left and right panes.

Your page should look like this:



Links (H2)

DotNetNuke® Site

Discussion Forum

Defect/Feature Tracking

Language Packs

Modules

Welcome To My Website... (H1)

Administrator Login: (H3)

Username: admin
Password: admin

* Select the Admin tab to manage the portal settings as the Administrator. Make sure you change the default password and specify a valid Email address for this account.

Host Login: (H3)

Username: host
Password: host

* Select the Host tab to manage multiple portals as the Host. Make sure you change the default password and specify a valid Email address for this account.

Sponsors (H2)

Books (H2)

Get this book from Amazon.com, written by members of the DotNetNuke® Core Team. Shaun Walker, Patrick Santry, Joe Brinkman, Scott

Further Info (H2)

Open Source License: (H3)

DotNetNuke® - http://www.dotnetnuke.com
Copyright (c) 2002-2005
by Perpetual Motion Interactive Systems Inc. (http://www.perpetualmotion.ca)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

This is the complete CSS code that we have added to the skin.css file:

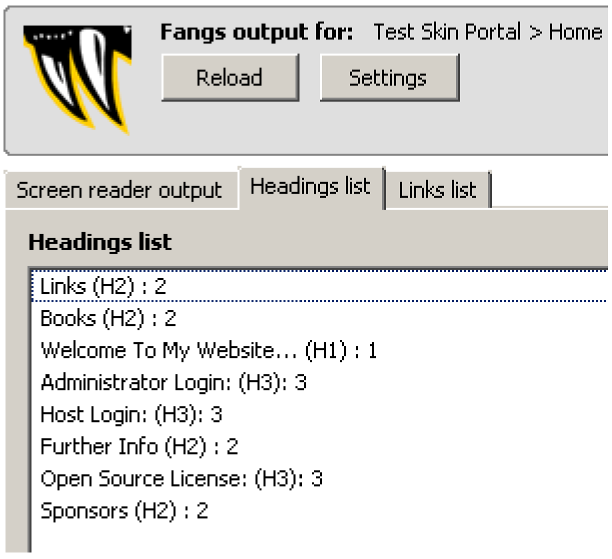
```
h1, h2, h3, h4
{
    margin: 0;
}

h1, h2
{
    padding-left: 3px; /* to replace the &nbsp; in the container file
*/
}

/* head class within the h1 tag */
h1 .Head {
    font-family: Tahoma, Arial, Helvetica;
    font-size:  18px;
    font-weight:   normal;
    letter-spacing: 0.2em; /* demo to show you can have different
text formatting  */
}

h3, h4 {
    font-family: Tahoma, Arial, Helvetica;
    font-size:  11px;
    font-weight:    bold;
    color: #000;
    margin-bottom: 10px;
}
```

Demo of Fangs

If we now view the Home page using:

Fangs for Firefox: The Screen Reader Emulator
http://www.standards-schmandards.com/index.php?show/fangs

Fangs Headings List



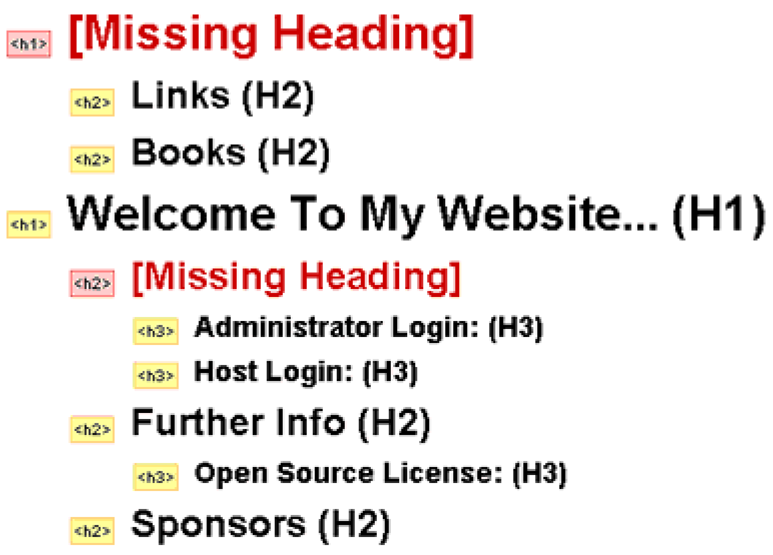
Headings list

```
Links (H2) : 2
Books (H2) : 2
Welcome To My Website... (H1) : 1
Administrator Login: (H3): 3
Host Login: (H3): 3
Further Info (H2) : 2
Open Source License: (H3): 3
Sponsors (H2) : 2
```

You will see that Fangs can read and display the list of header titles within our Home page.

Demo of Document Outline

Using the Firefox web developers toolbar, go to Information / View Document Outline.



<h1> [Missing Heading]

<h2> Links (H2)

<h2> Books (H2)

<h1> Welcome To My Website... (H1)

<h2> [Missing Heading]

<h3> Administrator Login: (H3)

<h3> Host Login: (H3)

<h2> Further Info (H2)

<h3> Open Source License: (H3)

<h2> Sponsors (H2)

You will see that the document outline view states that there are missing headings. This is because we have used the heading tags to outline the importance of the content, rather than placed the heading tags in the order of h1, h2, h3, h4, h5, h6 on the page.

If you view the websites that we analysed at the beginning of the tutorial, you will also see that some of them also receive missing heading statements.

Missing heading statements can be prevented when you design your actual skin. A missing heading statement is caused because the document outline is created by the order of the source code.

In the DNN-Blue skin, the order of the source code consists of the left pane, content pane and right pane.

This means that the first header tag that is read is a h2 tag, rather than a h1 tag and therefore the reader displays a missing header statement.

This doesn't cause any error problems, but it is something to be aware of as the most important element for your skin design is to have the important content (which should contain a h1 tag) appear at the top of the source code.

We discuss the order of source code in the following tutorials:
How to create a skin for DotNetNuke
Table tricks for search engine optimized skins in DotNetNuke
How to create a pure CSS skin

Final Thoughts

Ideally, to fully take advantage of pure semantic code, header tags, and the main content appearing at the top of the source code, you need to create a CSS based skin.

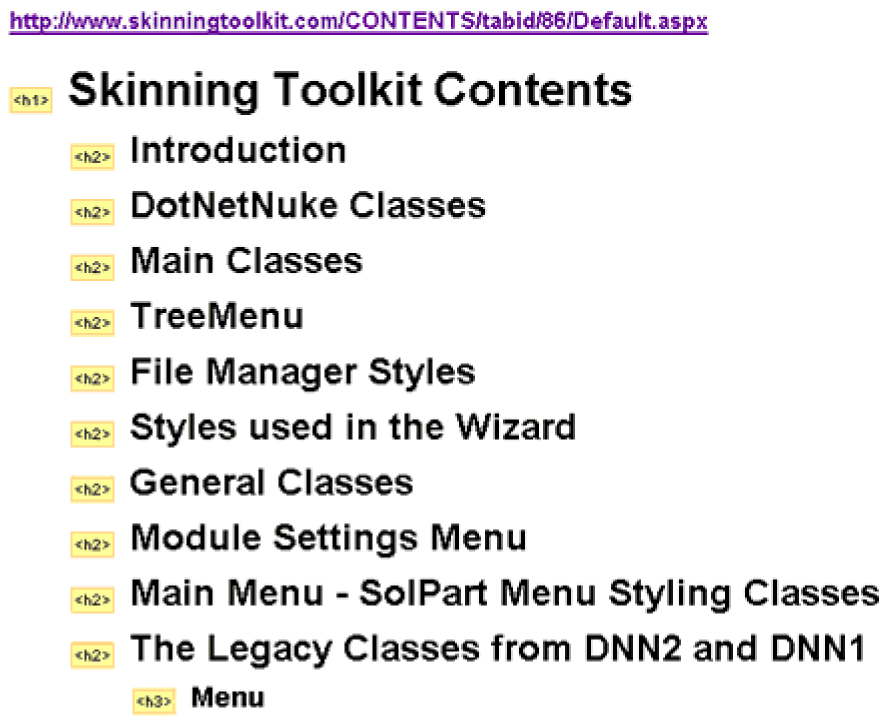
A semantic CSS skin using header tags will make your website more accessible to the variety of browsers such as screen readers, as well as give you an advantage within the search engines.

The methods demonstrated in this tutorial for applying header tags to a table based skin can also be used for a CSS based skin.

To view an example of a DotNetNuke CSS skin using a header tag structure as we have discussed in this tutorial, go to:
http://www.skinningtoolkit.com/CONTENTS/tabid/86/Default.aspx

and view the page using Fangs and the Web Developers Toolbar:

The Document Structure View


http://www.skinningtoolkit.com/CONTENTS/tabid/86/Default.aspx

<h1> Skinning Toolkit Contents
<h2> Introduction
<h2> DotNetNuke Classes
<h2> Main Classes
<h2> TreeMenu
<h2> File Manager Styles
<h2> Styles used in the Wizard
<h2> General Classes
<h2> Module Settings Menu
<h2> Main Menu - SolPart Menu Styling Classes
<h2> The Legacy Classes from DNN2 and DNN1
<h3> Menu

References / Useful links

7.5.5 Headings: The H1, H2, H3, H4, H5, H6 elements:
http://www.w3.org/TR/html401/struct/global.html#h-7.5.5

Semantic Code:
http://www.maxdesign.com.au/presentation/sit2003/05.htm

Semantic Code: What? Why? How?
http://www.boagworld.com/archives/2005/11/semantic_code_what_why_how.html

Why tables for layout is stupid: problems defined, solutions offered
http://www.hotdesign.com/seybold/everything.html

Books

Wrox, Professional CSS Cascading Style Sheets for Web Design by Christopher Schmitt, Mark Trammell, Ethan Marcotte, Dunstan Orchard, Todd Dominey. Published by Wiley Publishing Inc.