School of Computing, Maths & Digital Technology
Division of Digital Media & Entertainment Technology

Unit 6G4Z2101: Introduction to Web Design and Development

Worksheet 2: INTRODUCTION TO HTML

Workshop learning goals:

- Understand and apply common HTML tags
- Include hyperlinks and images in a web page
- Create HTML lists & tables
- Understand and apply new HTML5 structural tags

You should aim to complete at least pages 1-7 during the lab, and complete the rest before the following week's lab.

This worksheet will introduce you to the basic HTML syntax. If you are not familiar with HTML you can use online resources to help understand the code. If you need help, ask your lab tutor.

Some recommended resources for this week's worksheet are shown below, but there are many other resources available. The code included in the worksheet is correct for html5 rules.

http://www.lynda.com/HTML-tutorials/HTML-Essential-Training/170427-2.html

http://www.w3schools.com/tags/

https://docs.webplatform.org/wiki/guides/the_basics_of_html

HTML page structure

The code below shows the basic tags that begin the main structure of every web page.

```
1   <!DOCTYPE html>
2
3   <html lang="en">
4
5   <head>
6       <meta charset="utf-8">
7
8       <title>Title text goes here</title>
9
10  </head>
11
12  <body>
13  Page content goes here
14  </body>
15
16  </html>
17
```

Getting Started with HTML

To begin writing html use Notepad++ (PC) or Textwranger (Mac).

NOTE: text editors which include formatting information (eg Microsoft Word) should **not** be used for authoring web pages. Any plain text editor will do.

In order to preview your HTML page you will need a web browser, such as Chrome, Firefox, Internet Exploer, etc. You do not need to be online.

IMPORTANT: Always save an html file with the file extension "**.html**" or "**.htm**". Notepad will default to a ".txt" file extension. When you save your file, you can override the default by enclosing the filename text in speech marks e.g. "*your_file_name*.**html**". You can also override it by selecting "All Files" in "Save as Type".

Set up your folders and create an html document

1. Create a folder for this unit in your own network area (H:drive).
2. Create another folder inside the unit folder called 'lab1'.
3. Using a text editor create a new document and save it into your 'lab1' folder with the name "**index.htm**".
4. Add the block of code shown below, replacing "*Title text goes here*" with your name and "*Page content goes here"* with a few sentences, spacing them out so there is a blank line between them.

```
1   <!DOCTYPE html>
2
3   <html lang="en">
4
5   <head>
6       <meta charset="utf-8">
7
8       <title>Title text goes here</title>
9
10  </head>
11
12  <body>
13  Page content goes here
14  </body>
15
16  </html>
17
```

5. Save your document and preview in Chrome (in Notepad++ choose Run > Preview in Chrome) and note the following:
   a. Where does your name (i.e. the <title>) appear on the web page?
   b. Why is the layout of the sentences different to how they appear in the text editor (i.e. no blank lines)? We'll look at this shortly.

DOCTYPE and Attributes

Using your preferred search engine, find out the purpose of the following statements in your html document, and explain your understanding of them in your own words. Find out how the DOCTYPE from XHTML to HTML5. Make a note of your findings.

- <!DOCTYPE html>
- lang="en" (which is an attribute of <html>)
- <meta charset="utf-8" (charset is an attribute of <meta>)

Paragraphs and Headings

1. Refer to http://www.w3schools.com/html/html_basic.asp to understand how the paragraph tags and different levels of heading e.g. <h1>, <h2>, etc

2. Add paragraph tags to your text as appropriate to space your sentences out when your page is displayed in a browser.

3. Add the following code to your page, save it, and preview it in a browser. Note the difference between the size of text for <h1> and <h2>. You can use heading levels up to <h6>.

```
<h1>Heading level 1</h1>

<h2>Heading level 2</h2>
```

If you want additional white space between your paragraphs, you can use the break tag **
**.

The
 tag

The break tag
 will add additional white space between paragraphs where you want to space your paragraphs out more. Note that if you have been used to using
, in html5 you no longer need the closing /.

1. Add a
 tag between a couple of your paragraphs then save and view your page in a browser. Note the effect of
.

Adding images

Images are added to a web page using the 'img' tag, in this format:

<img src="*image name with file extension*">

e.g. <img src="logo.jpg">

If the image is stored in an 'images' folder, you also need to point to the folder.

e.g. <img src="**images**/logo.jpg">

1. Create an 'images' folder inside your 'lab1' root folder, then find an image to add to your page (e.g. you, your favourite band, film character, football team, etc), and save it to the new folder.
2. Referring to the code above, display your image in your web page. Save the page and check it in a browser to make sure you can see it. If you see a red x where the image should be, check your file name, file extension, and location of the image.

Image Attributes

The <img> tag can have some attributes associated with it. You can add 'alt' text, 'width' and 'height'. For example:

<img src="images/logo.jpg" alt="Company logo" width="200" height="200">

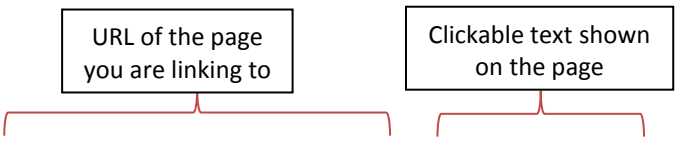
Width and height use pixels by default as their unit of measure. By setting them within the <img> tag, the space required to display the image is reserved when the page is loaded by the browser. This can help to speed up load time and helps to avoid the layout changing as the page renders.

The alt text is there for the browser to display if it is unable to display the image, or for users using screen readers (which will be discussed in a future lecture).

Adding hyperlinks

As a network of inter-connected content, hyperlinks are an essential element of the World Wide Web. They are created using anchor tags. You can create **internal** links to other pages within your own site, or **external** links to pages outside of your own site.

The anchor tag below is an **external** link to the W3 schools website. Note that the link includes the full path name, including http://. Note the tags involved.



Visit W3Schools

The anchor tag below is an **internal** link to the W3 schools website. Note that it uses only the filename and extension. This assumes that it is stored in the same folder as the page you are linking from.

Link to my file

1. Create a second html page and store it in the same folder as your current page.
2. Add an internal link to each of the pages so that you can click from one page to the other.
3. Add a couple of **external** hyperlinks to each page. You could use some of the links included in this worksheet.
4. Test to see that all your links work.
5. Find out what 'href' stands for (e.g. using the W3Schools website) and make a note of it.

HTML lists

It is important to understand how lists work in html, because navigation menus are commonly created using styled lists.

1. Create a new html page with all the standard tags and save it to your current folder as **lists.htm**.
2. Give your page the <title> "HTML lists".
3. Watch the following two tutorials on lynda.com explaining how to create **unordered** (i.e. bulletted) and **ordered** (i.e. numbered) lists in html. Reproduce the lists shown below. Note that to create the indented 'sub-lists', you **nest** one list inside another.

   http://www.lynda.com/HTML-tutorials/Unordered-lists/170427/196172-4.html

   http://www.lynda.com/HTML-tutorials/Ordered-lists/170427/196173-4.html

Ordered List

1. First item
2. Second item
    1. Sub item
    2. Sub item
3. Third item
    1. Sub item
    2. Sub item

Unordered List

- Bullet item 1
- Bullet item 2
    - Sub item 2.1
    - Sub item 2.2
    - Sub item 2.3
- Bullet item 3

Mixed List

1. Numbered item 1
2. Numbered item 2
    - Bullet 1
    - Bullet 2
    - Bullet 3
3. Numbered item 3

HTML tables

Historically, tables were used to control the layout of a web page. This is no longer acceptable within web standards, and layout should be controlled using stylesheets (next week's topic). Tables should now be used only to display data.

Attached is a page showing some html tables.

1. Create a new web page with the basic tags, and save it to your root folder as **tables.htm**.

2. Refer to the following web page for information on how to create html tables, and re-create the tables shown on the next page. There are some links on the next page to some online resources.

   http://www.tutorialspoint.com/html/html_tables.htm

   The main points to note are:

   - <table> </table> open and close the ***table***
   - <tr> </tr> open and close a ***row***
   - <td> </td> open and close a ***cell*** (table data)
   - <th> </th> add a table ***header*** (which appears in bold)
   - <caption> </caption> adds a ***caption,*** which appears above the table
   - If you want a ***border***, you need to define one (e.g. border = "1" will create a 1px border around the table and its cells)
   - Cells can be merged using ***rowspan*** and ***colspan***

NOTE: If you want to put some white space between your tables, you can use the **
**. Tag.

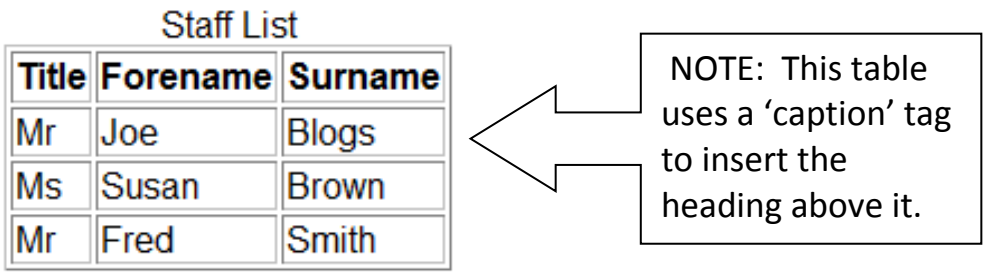
Some additional resources can be found at:

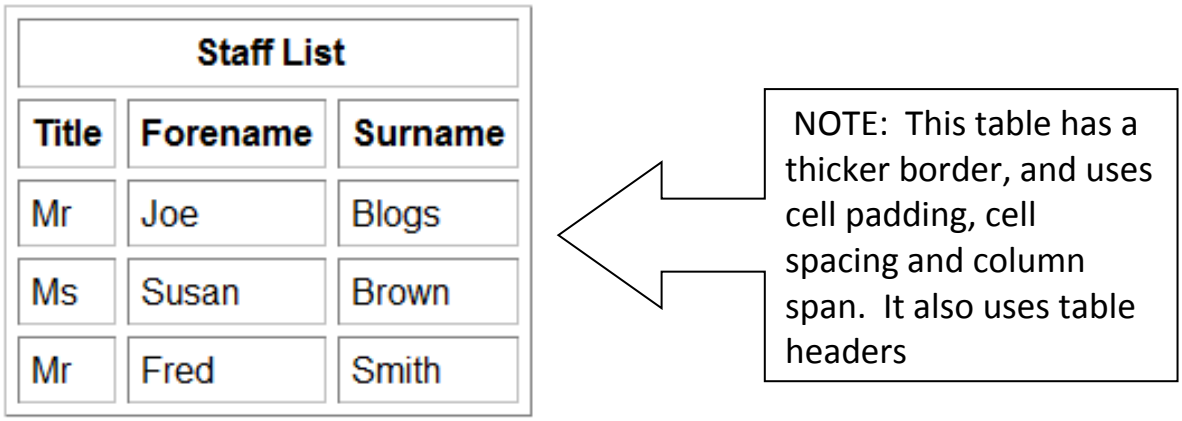
http://stackoverflow.com/questions/9830506/how-do-you-use-colspan-and-rowspan-in-html-tables

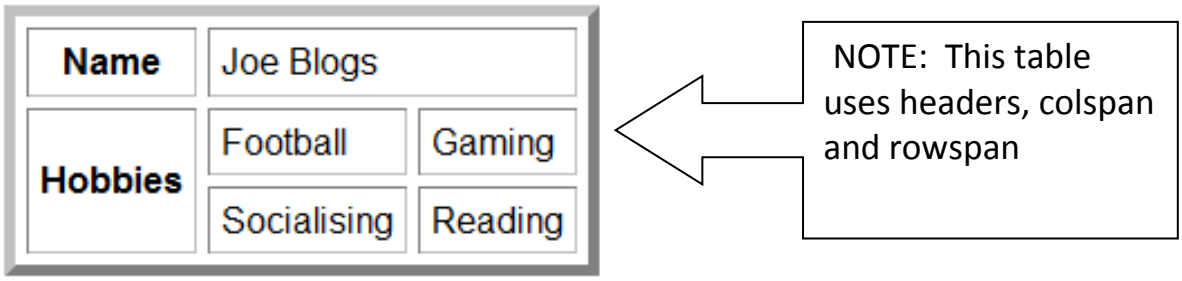
http://www.htmlcodetutorial.com/tables/index_famsupp_30.html

http://www.htmlite.com/lite012e.php

http://www.tizag.com/htmlT/tables.php







Tutorials for developing your understanding

If you are not completely familiar, from prior experience, with the content covered in this worksheet you should view the following Lynda.com tutorials to develop your understanding.

1. Go to the link below and watch the two tutorial sets listed. You can make your own notes within Lynda.com, download exercise files, and practice your code.

   http://www.lynda.com/HTML-tutorials/HTML-Essential-Training/170427-2.html

   - Introducing HTML
   - Basic Page Structure

2. Review what you have done then create a new html page and try to add all the basic tags from memory. Include at least one headings and a couple of paragraphs. To see if you have remembered everything, try validating your page using the W3Cs Markup Validation Service. Choose the 'Validate by File Upload' tab.

   https://validator.w3.org/#validate_by_upload

3. Choose 4 of the links in this worksheet, then add them to your new html document as part of a bulleted list. Each link should be a separate list item.

4. Try validating your page again and fix any errors.

Creating a meaningful structure in your code

HTML5 introduced some new **semantic structural** markup tags which help to add standards to the structure of tags. In linguistics semantics are concerned with **meaning**. With web development, your html tags allow you to indicate the structure of documents. For example, think about different levels of heading tag indicating how important different headings are in relation to one another.

HTML5 introduced some new tags to help add meaning to the structure. They include <header>, <footer>, <nav>, <article>, <section> and <aside>. There is still some confusion about when to use <article> and <section>. Often it comes down to personal judgement as to which one is the more appropriate in any given circumstance.

<article>

The W3C specification for <article> is shown below.

> The article element represents a complete, or self-contained, composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

Essentially, an article is content which makes sense on its own, even if you were to remove it from all other content on the page.

Read the full specification at http://www.w3.org/TR/html5/sections.html#the-article-element

<section>

The W3C specification for <section> is shown below.

> The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content. The theme of each section should be identified, typically by including a heading (h1-h6 element) as a child of the section element.
>
> Examples of sections would be chapters, the various tabbed pages in a tabbed dialog box, or the numbered sections of a thesis. A Web site's home page could be split into sections for an introduction, news items, and contact information.
>
> Authors are encouraged to use the article element instead of the section element when it would make sense to syndicate the contents of the element.

So a section can be seen as sitting within an article, being related to the article, and being worthy of its own heading.

Read the full specification at http://www.w3.org/TR/html5/sections.html#the-section-element

<aside>

The W3C specification for <aside> is shown below.

> The aside element represents a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography.
>
> The element can be used for typographical effects like pull quotes or sidebars, for advertising, for groups of nav elements, and for other content that is considered separate from the main content of the page.

So content in an <aside> is essentially additional information related to the <article> or <section> it is nested with. If you removed an <aside> it would not detract from the meaning of the content to which it is related. A sidebar on a web page may be considered as an <aside>, but this may depend in its content, and its relevance to the other content surrounding it.

In very simple terms, you could think of a book as an article. It is a complete composition, and is worthy of its own title.

Within a book you could consider each chapter as a section. Chapters again are significant enough to have their own headings. A single chapter on its own, out of the context of the other chapters, may not be completely meaningful, so needs to be grouped with the other chapters for complete meaning. This is what the specification refers to when it talks about syndication. Because it needs the other chapters in order to be fully meaningful, a chapter is better defined as <section> rather than <article>.

An example of where an aside might be relevant within the example of a book is if there were some footnotes, or further references, which provide additional but not essential information.

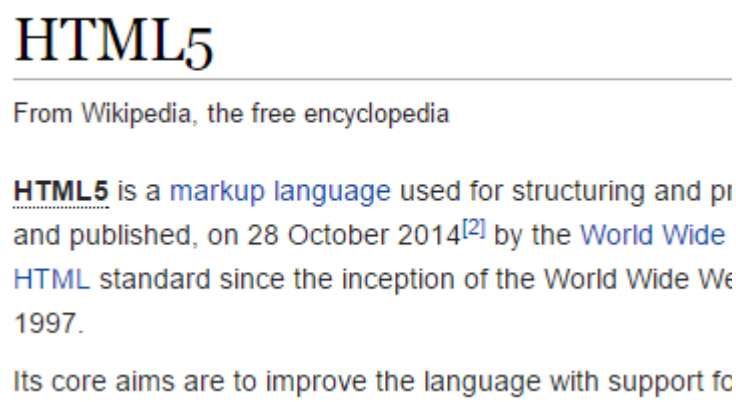
So what happens if a chapter has sub-sections which have their own headings?

To some extent this is where things can get a little blurred, and where you may have to use personal judgement. The essential thing is that you maintain a consistent approach.

Look at the following Wikipedia page about HTML5 (there are much better sites for information than Wikipedia but the layout of their pages is convenient for illustrating these semantic tags).

https://en.wikipedia.org/wiki/HTML5

At the top of the page is a main heading - "HTML" - followed by some introductory text.



HTML5

From Wikipedia, the free encyclopedia

HTML5 is a markup language used for structuring and pr
and published, on 28 October 2014[2] by the World Wide
HTML standard since the inception of the World Wide We
1997.

Its core aims are to improve the language with support fo

Then we have the contents.

Contents [hide]

1 History
- 1.1 Standardization process

2 Features
- 2.1 Markup
- 2.2 New APIs
- 2.3 XHTML5 (XML-serialized HTML5)
- 2.4 Error handling
- 2.5 Popularity
- 2.6 Differences from HTML 4.01 and XHTML 1.x

3 Logo
4 Digital rights management
5 See also
6 References
7 External links

Within 'Contents' there are some main headings and some sub-headings.

In terms of semantic tags, we could consider the entire page as an **<article>** about HTML5.

Each of the blocks with main headings within the <article> can each be considered as a **<section>**, each being related to the topic of the article.

In semantic coding terms, the "See also" content might best be considered as an **<aside>** rather than a <section> or <article>. The information in this part of the document might be useful, but it is not essential, and refers to content which is from external sources.

Think about how you might treat "References" and "External links" in terms of these semantic structural tags.

<header>

In addition to the structural tags described above, HTML5 also offers a <header> and <footer> tag. Their use can vary in the sense that <header> could be the header for a page, or the header for an article within the page, or a section.

The W3C specification for <header> is shown below.

> The header element represents introductory content for its nearest ancestor sectioning content or sectioning root element. A header typically contains a group of introductory or navigational aids.
>
> When the nearest ancestor sectioning content or sectioning root element is the body element, then it applies to the whole page.
>
> A header element is intended to usually contain the section's heading (an h1–h6 element), but this is not required. The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos.

Read the full specification at http://www.w3.org/TR/html5/sections.html#the-header-element

<footer>

The W3C specification for <footer> is shown below.

> The footer element represents a footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like.
>
> When the footer element contains entire sections, they represent appendices, indexes, long colophons, verbose license agreements, and other such content.

Read the full specification at http://www.w3.org/TR/html5/sections.html#the-footer-element

Like the <header>, any given page could have more than one set of <footer> tags.

See the example structure diagram under “Basic HTML5 Structure” on the following web page http://www.basewebmaster.com/html/html5-page-structure.php

In the Wikipedia example above, the introduction which appears before the list of contents could be tagged as a <header>.

<nav>

The W3C specification for <nav> is shown below.

> The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links.
>
> In cases where the content of a nav element represents a list of items, use list markup to aid understanding and navigation.
>
> Not all groups of links on a page need to be in a nav element — the element is primarily intended for sections that consist of major navigation blocks.

Read the full specification at http://www.w3.org/TR/html5/sections.html#the-nav-element

You can learn more about the above tags, and see examples of their use in the following Lynda.com tutorials.

- See section 4. Structuring Content at:
  http://www.lynda.com/HTML-tutorials/HTML-Essential-Training/170427-2.html

- See section 1. Using HTML5 Semantic Tags at:
  http://www.lynda.com/Web-Content-Strategy-tutorials/Web-Semantics/143180-2.html

Self-directed learning task

The following activities require you to illustrate your understanding of HTML tags, and semantic structure. It should be completed on your own, in your own time, before your next lab session.

1. Look carefully at the content on the following pages.

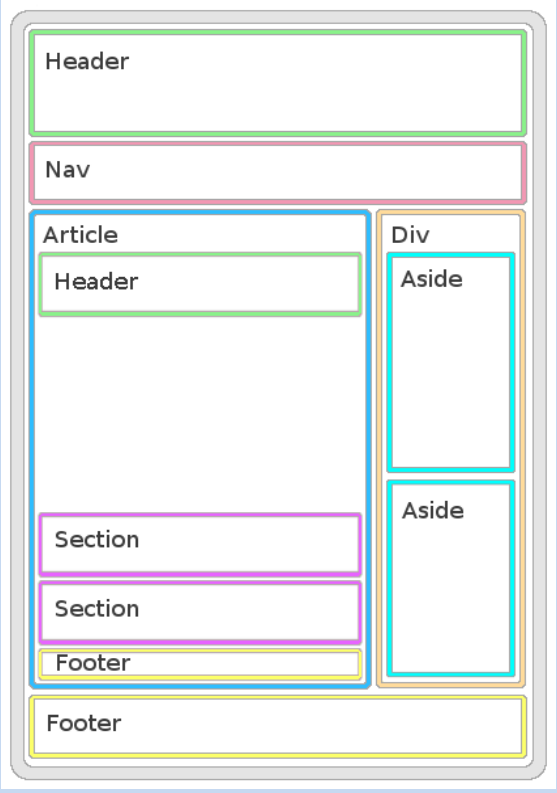
   Consider how you would tag it up. Consider levels of heading, paragraphs, and the structural elements described above.

   Create a new HTML document and type out your complete tag structure. You do not need to include all the content, but do include a basic indication of content where appropriate. For example, the first couple of paragraphs could be shown exactly as displayed below.

   As HTML5 is now …
   Effectively the basic …

2. Look at the illustration of the page structure shown on the right and add to your HTML document the tags required to reproduce it.

   You will need to think about how tags are nested in order to have blocks of content positioned inside other content blocks.



20 Feb 2011

The HTML5 Page Structure

HOW TO STRUCTURE PAGES USING THE NEW HTML5 ELEMENTS

Html5 • Structure • Layout

As **HTML5** is now being implemented by webmasters all around the web I thought it only appropriate to cover some of the new elements and page structure basics here at Basewebmaster. Particularly as many of the websites demonstrating the HTML5 structure appear to have *misunderstood* how the new structural elements are used.

Effectively the basic structure of a HTML5 document has not changed. Each comprises of a head section containing unseen details and links and a body section where the visible elements of the document reside.

Firstly lets take a quick look at the head section code shown below.

Table of Contents

HTML5 Document

```
<!DOCTYPE html>
<html lang="en-US">

<head>

<meta charset="utf-8" />
<meta name="keywords" content="key, words" />
<meta name="description" content="description" />

<link rel="stylesheet" href="stylesheet.css" type="text/css" />
<link rel="alternate" title="Website Feed" href="rss.php" type="application/rss+
<link rel="icon" href="favicon.ico" type="image/x-icon" />

<title>Page Title</title>

</head>
```

DOCTYPE

The doctype as you can see is much simplified in HTML5 as there is only one type of HTML document it is simply defined as **<!DOCTYPE html>** simple as that. Nothing more is needed to define the doctype.

HTML Element

We now need to open the html element <html lang="en-US"> notice that the language attribute is added to the html tag. This is adequate to show that the whole document is written in English so the language attribute does not need to be added to each element. The exception would be if you whished to show and alternate language within the body of the document.

For example if you were to provide links to the document in alternate languages you may wish to show those links in the native language and therefore the lang attribute should be applied to that element and would override the document root language.

The html tag is also the place to add the dir attribute if needed, either ltr or rtl. By default ltr is used so it is down to your own usage whether you define this attribute or not.

Head Element

Now we open the head section of the document. Much remains unchanged; in fact we have not added any elements to the head section simple removed unnecessary ones.

Meta Data

We have no need for any meta data relating to content type as we have already define the doctype to HTML5. Neither do we need any language declaration as we have added that to the html base element. The only metadata that is recommended is a charset declaration, meta keywords and meta description as shown in the example.

Links / Scripts

In terms of any link or script elements nothing much has changed regarding the way they are shown, just make sure they each have the necessary rel attribute and type attribute attached.

Finally the title element remains unchanged and functions as it always has. That is the head element, of course there are many other things you can add to the head section particularly different link elements used by various search engines and user agents. For the purpose of HTML5 however the above is the basic structure.

The Body Element

Now on to the body element, all content i.e. visible parts of the document should be placed within the body element; this at least has not changed in HTML5. The main modification to the body element is with the addition of a number of new elements unique to HTML5 and the way the page is effectively divided into **sections**.

They have been added to make the mark-up of a document easier and supposedly more intuitive. So instead of having loads of div and span elements you will now have a more appropriately named element to contain various type of content.

Okay so first of all lets take a look at the new elements along with their technical definition and defined place or place's of usage.

header

The header element represents a group of introductory or navigational aids.

- Header elements relate to the sections under which they occur, if they occur as direct descendants of the body element then they apply to the document as a whole.
- They are **not** limited to h1-h6 elements in fact they can contain almost any content you wish as long as it is appropriate heading data.
- A document may contain as many headers as you like but they should be limited to one per sectional container.

nav

The nav element represents a section of a page that links to other pages or to parts within the page: a section with navigation links.

- Nav elements relate to the sections under which they occur, if they occur as direct descendants of the body element then they apply to the document as a whole.
- They may contain heading and text as well as links.
- Navigational elements should be used to identify large blocks of navigational data, they are not needed for smaller navigational displays.

article

The article element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication. This could be a forum post, a magazine or newspaper article, a blog entry, a user-submitted comment, an interactive widget or gadget, or any other independent item of content.

- Article elements contain the documents or page specific content of that specific document. For a blog the article tags would surround the blog post itself. For a forum the article would comprise of the initial post with subsequent posts / replies nested within it.
- Multiple article element **are** allowed either nested within one another or independant from each other. If independant from each other this would imply that the article elements contain content which would be self supporting i.e. content contained within the other article elements is not neccessary for understanding.

section

The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading.

- Section elements are used to group similar content into a block, as such all content within a section element should be directly related.
- Section elements are considered as seperate sections of a document and therefore should contain their own header data.
- Should **not generally** be used to contain the **main body** of a document as this is what the article element is for, may be used to break up a document into topical sections.
- Best used as a **container** within an article element to group common themed content e.g. The comments section of a blog should occur within the main article element but would best be contained within a specific section element with an appropriate heading such as comments.

aside

The aside element represents a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content. Such sections are often represented as sidebars in printed typography.

- Aside elements should contain content which is very loosely or unrelated to the main theme of the element in which they occur and the elements arround them.
- Within a blog aside elements may for example contain the side bar links to various other blog posts.

footer

The footer element represents a footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like.

- Footer element should be used based on what they will contain not where abouts within a documnet they occur. Footer elements do not need to be at the foot of a document they simple need to contain appropriate information about the document or section.
- multiple footers are allowed, one for each sectional container is acceptable.
- For a blog page the contact and copy write data at the bottom of the page would be within a footer. Also the author and date published for the blog entries, often positioned just beneath the blog heading would also be a footer.