Lab 02 – Implementing Wireframes in HTML

The aim of this lab is to make you more aware of the information architecture and structure of some webpages you may already use; become familiar with basic HTML5 syntax to model the structure and content of a blog-style web page; and apply these skills in implementing a simple interface wireframe in HTML5.

### Task One – Visualising Page Architecture

* In Firefox, open **Tools/Web Developer**

1. Navigate to the **Tools/Web Developer** menu
2. Select **Inspect**
3. Experiment with this tool for analysing a page
   1. What information does it provide?
4. Select the **3D View** button
5. Experiment with this view
   1. What information does this view provide?

*Note: if you would rather use chrome you can do the same with the following extension https://chrome.google.com/webstore/detail/3d-view-for-webpages/noflnemoniodcoagmaapcdlmeljohdnm?hl=en*

* Analyse the following web pages (you may have to navigate or sign in to find the ‘main’ interface of each website:
  + [www.google.com](http://www.google.com)
  + [www.flickr.com](http://www.flickr.com)
  + [www.reddit.com](http://www.reddit.com)
* Describe some of your observations in a new Google doc by answering these questions:
  + Which HTML elements occur most often for each page?
  + Is there a relationship between the ‘depth’ of the 3D visualisation and the complexity of the web site appearance?
  + Does the 3D visualisation help your understanding of web pages being complicated data-structures and not simply a 2D page?
  + [www.facebook.com](http://www.facebook.com)

### Task Two – Basic HTML Primer

The following wireframe describes the design of a simple blog. There are four main sections:

* **PageHeader**: contains the Site*Logo* and *SiteName*
* **SiteNavigation**: contains a list of navigation links (*Home*, *About*, *Contact*)
* **PageContent**: contains one or more *BlogPosts*. Each *BlogPost* contains a *BlogPostTitle*, *BlogPostDate* and *BlogPostContent*.
* **PageFooter**: contains a link to *LegalStuff*

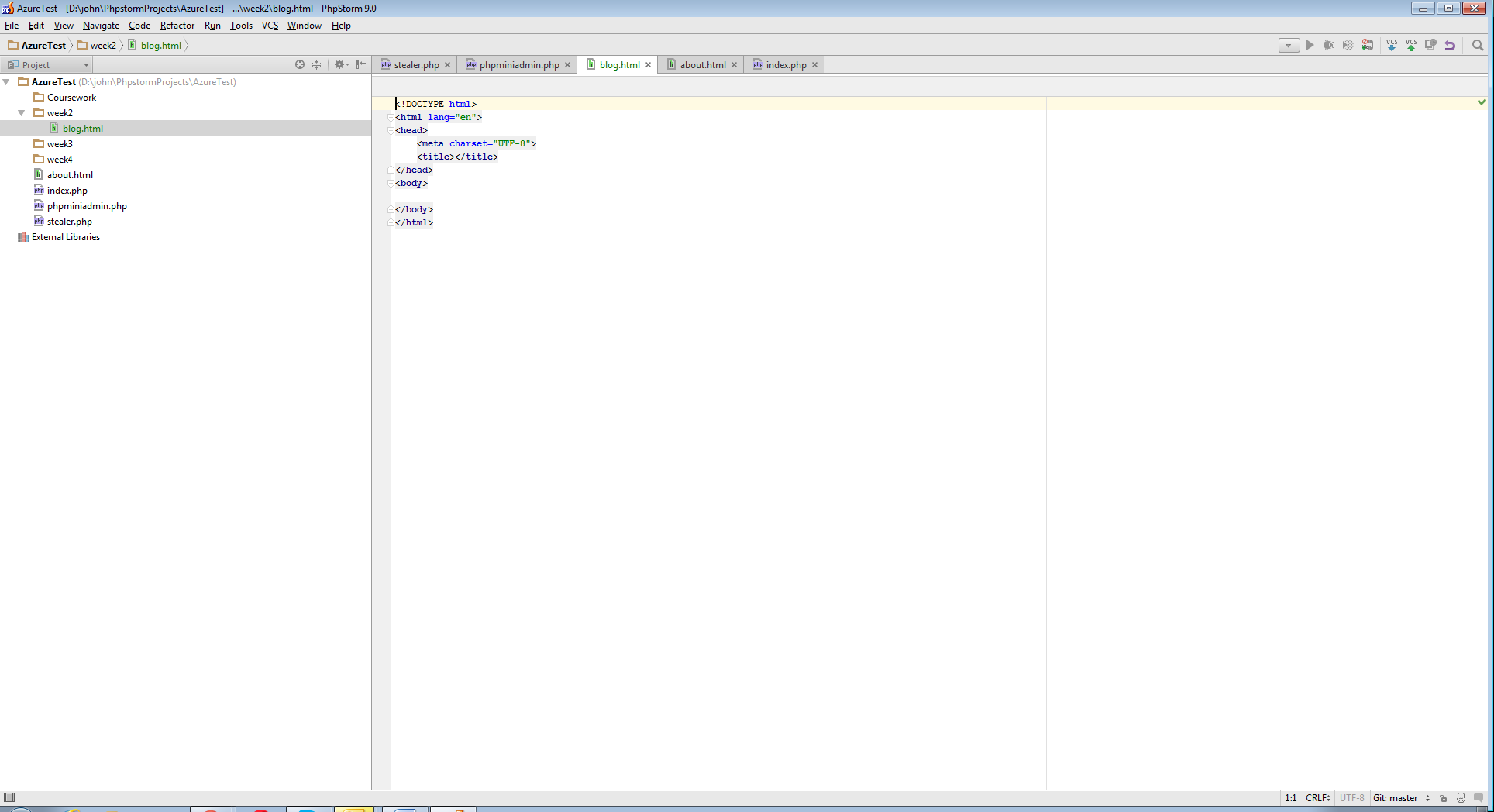
The layout of these main sections is in a single column.

blog-architecture.pdf

### Part A – Create working project and blog.html file

Last week we set up our PHPStorm and our Azure hosting. When you set up GitHub and Azure you tied the phpstorm project to your GitHub repository. So everything you commit/push to GitHub will be live on your Azure site.

For this module it is best if you use a **single project** and create **subfolders** for each week’s code.



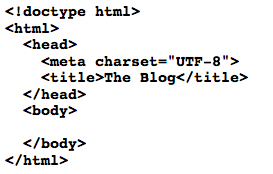
So once you have done **rightclick >git>commit** to push your code to Azure.

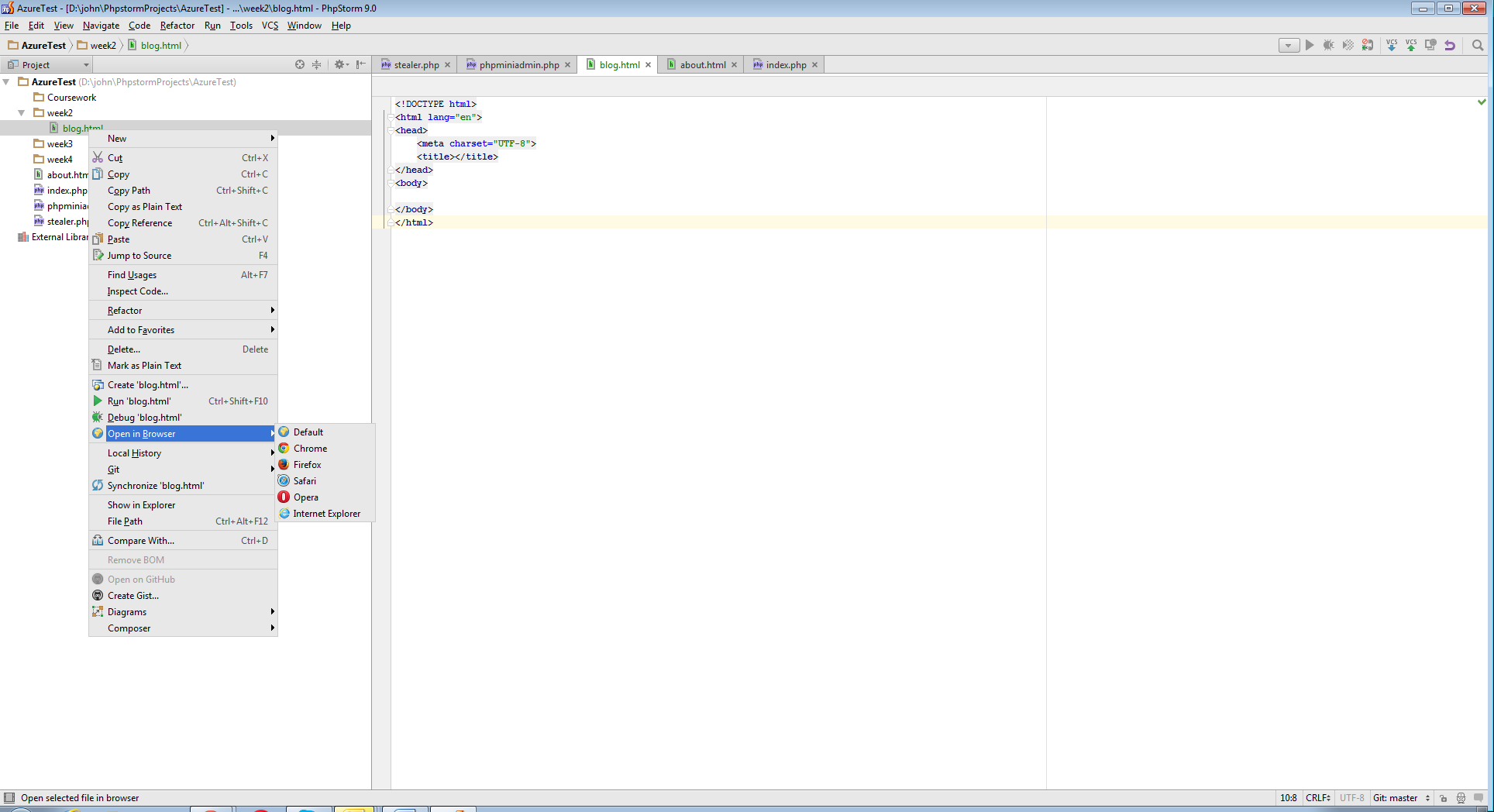
If you wanted to access **blog.html**, you just enter **<yoursite>.azurewebsites.com/week2/blog.html** in your browser.

* Create a week2 folder in phpstorm
  + *(rightclick project>new>directory)*
* create a new HTML file called blog.html
  + *(rightclick your directory > new> html.file)*

### Part B – Create the basic <head> and <body> elements of the page

Add the following HTML code to **blog.html**:





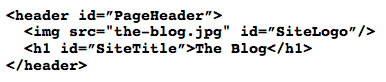
You can view you file locally using the **open in browser tool. (it’s best to use firefox in the lab)**

It should be empty, except for a page title “The Blog” appearing in the current tab. Keep reloading blog.html after each part.

### Part C – Create PageHeader with <header>, <img>, <h1> tags

The PageHeader contains our blog’s name and logo. To create the PageHeader, we can use the **<header>** tag. Note that an HTML page can have multiple headers, e.g. a header for a single BlogPost. To help distinguish them, we can include an **id attribute**. It is essential practice to use id and class attributes, so make sure you do so where appropriate.

Add the following code in-between the **<body> … </body>** tags:

****

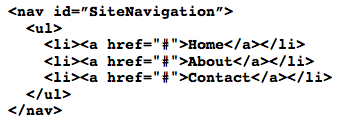
You will have to find a suitable image for your logo. Save it to the same directory as blog.html.

### Part D – Create SiteNavigation with <nav>, <ul>, <li>, <a> tags

The SiteNavigation provides an unordered list **<ul>** of links **<a>** to other parts of our blog.

For now these links will not go anywhere **src=”#”**. Whilst using **<div>** or **<section>** tags to contain navigation links is perfectly legal HTML, we can use the **<nav>** tag to be more clear about the meaning and purpose of this section. As before, there can be multiple **<nav>** tags, so we add an **id attribute**.

Add the following HTML code after the PageHeader section.

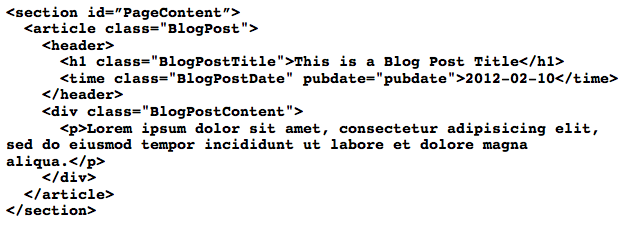
****

### Part E – Create PageContent and BlogPost with <section>, <article>, <div>, <time> and <p> tags

The PageContent contains the main content of the blog, its BlogPosts. To wrap one or more BlogPosts, we can use the **<section>** tag. Once more there are not any strict rules in HTML about which tag is used, but the **<article>** tag makes sense to model a BlogPost.

Instead of an **id attribute**, the **class attribute** is used instead for BlogPosts. This should be obvious as a blog typically has more than one blog post. So, rather than identify an individual element in our HTML, the class attribute allows us to refer to a **set of similar elements**.

The **<article>** tag for the BlogPost contains several elements to help model a typical blog post: **<h1>** for the BlogPostTitle and **<time>** for the BlogPostDate, both wrapped in a **<header>** tag and **<div>** to hold the actual BlogPostContent.

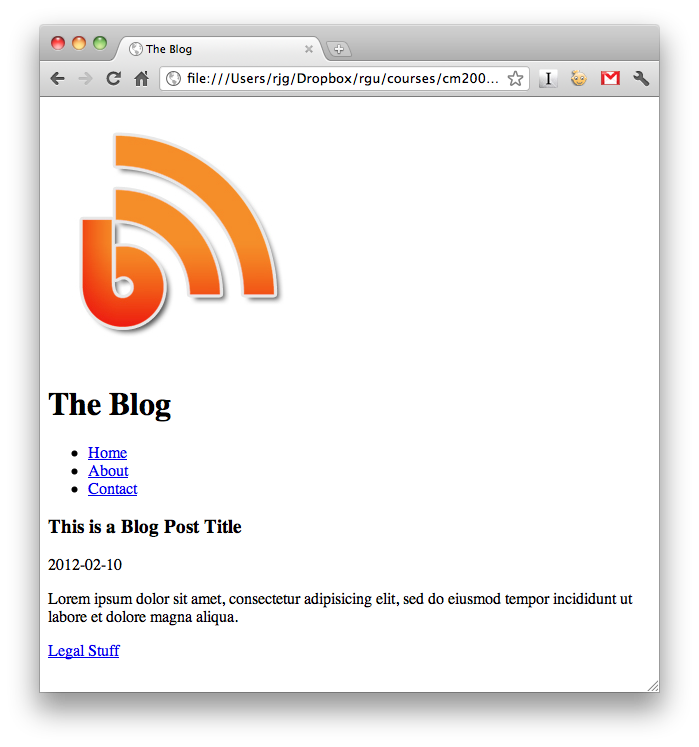
****

### Part F – Create PageFooter

Finally we can complete the page with a PageFooter (with links no one ever reads) modelled using the **<footer>** tag:

Macintosh HD:Users:richard:Desktop:Screen Shot 2013-02-07 at 11.33.59.png

And the result should be similar to this effort (remember to use Firefox’s 3D View via the Inspector to compare your page to the other pages viewed earlier):



Once completed commit and push your code to GitHub, you can then visit your site and check that your live code functions just like your local code. Also because your code is now on GitHub it is all nicely backed up!

### Task Three – Implement a Photo Gallery Wireframe

In this task you will implement the given wireframe in HTML5. Note that you may not be able to achieve the layout and positioning of the sections at this stage, but you should be able to create the logical structure and order.

You should make appropriate use of HTML5 elements (header, nav, section, footer, etc) when creating the wireframe. You should also create corresponding id and classes for the labels in the wireframe.



### Task Four – Implement your own Wireframe in HTML

In the lecture you drew some wireframes pick the best from your group and attempt to implement it. You may want to revise the earlier attempt to include more descriptive id’s and classes, as illustrated in Task Two wireframe. More complex layouts (2+ columns) may not be possible at present, so instead consider how the sections of the page flow and order them appropriately.

### In the remaining time…

Continue developing your web app concept with your group. Discuss the following questions:

What do we want to build?

What 3rd party data do we need from a web service (e.g. pictures / video, news, etc)?

What is our information architecture?

What is the **Context** of the web app?

Who are our **Users**?

What is the **Content**?