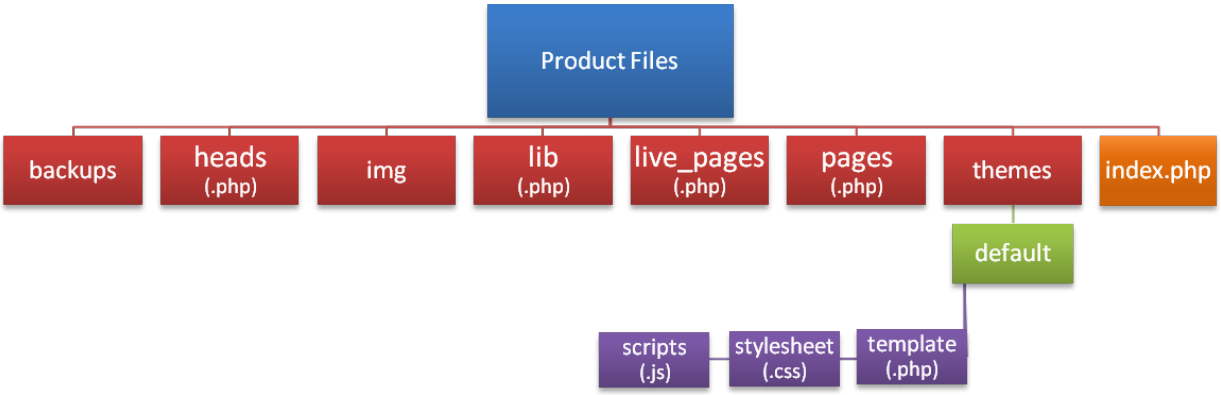
Product Development (12 Marks)

The following techniques were used to develop the website

- Use of Cascading Stylesheets (CSS)
- Use of Javascript to customize pages and improve functionality
- Use of PHP to customize pages and improve functionality
- Use of Ajax (Combining Javascript and PHP)
- Manipulated graphics

1

Organisation of the website (Document object model)



The following diagram represents the structure of the website:

Website content is placed appropriately in carefully named folders, e.g. the 'img' folder is used to store all of the images used for tutorials. The 'themes' folder layout allows for new themes to be created in the future.

Individual page content is held within the 'pages' folder, and the template files within the 'template' folder of the theme currently in use. The index.php file in the home directory is used to bring together the page content, templates, library classes and any additional information (stored in 'heads') being used.

The structure and organisation of webpages

Technique 1: Cascading Stylesheets (CSS)

Cascading Stylesheets are used in the presentation of a web page, and are designed to promote consistency across web pages (see success criterion 1) by allowing the creation of styles named as Classes or IDs, as well as the ability to stylise tags (objects) such as headings and bullet points. This eliminates the need of reproducing code across multiple pages and thus allows for faster page loading. Changing the presentation of headings within a stylesheet, for example, will affect every page that links to it – one major bonus of using stylesheets. Stylesheets are inserted in the page head (between the <head> and </head> tags), for example:

```
<link rel="stylesheet" href="default.css" type="text/css" />
```

This is the same for any stylesheet being used, with the 'href" attribute being changed to point to the appropriate stylesheet file (in my case default.css). Below is a screenshot of my stylesheet 'default.css':

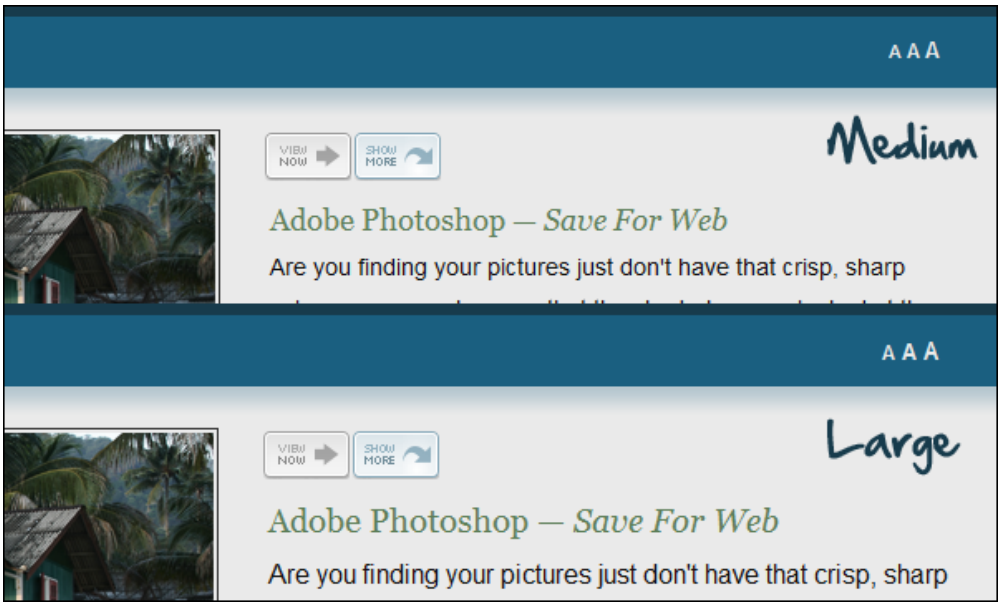
```
/* -=HTML ELEMENTS=- */
body {
    margin:0px;
    padding:0px;
    background:#EAEAEA;
    font-family:Arial, Helvetica, sans-serif;
    font-size:0.75em;
    line-height:2em;
    color:#111;
}

a { cursor:pointer; text-decoration:none; color:#173B4C; }
h1 { font-family:Georgia, "Times New Roman", Times, serif; font-weight:normal; m
h2, h3, label { font-family:"Trebuchet MS", Arial, Verdana; font-size:120%; font
h3 { font-size:110%; color:#5f9424; }
h3 a:hover { border-bottom:1px solid #173B4C; }
```

3

Technique 2: Javascript

Whilst also being used with Ajax (explained later), Javascript was used to satisfy my requirements (criteria 4 and 5) of making the website accessible to all users. I wanted to be able to allow the user to specify the font size they wanted to use (small, medium, or large), and was able to do this by using Javascript. With the help of jQuery, (http://jquery.com/, © 2009 John Resig and the jQuery Team), a Javascript library, I used Javascript to change font size dynamically. (below)



Using jQuery allowed me to simply use three lines of code (below) to change font size.

```
$("#resize_small").click(function() { $("body").css("font-size", "0.7em"); });

$("#resize_default").click(function() { $("body").css("font-size", "0.75em"); });

$("#resize_large").click(function() { $("body").css("font-size", "0.85em"); });
```

4

Technique 3: PHP

PHP is a *hypertext pre-processor* which means that it deals with information from databases and files etc. and processes it into HTML. For this reason, I used PHP to help bring together template files (from the 'template' folder) and the relevant page information (see success criteria 1 and 2). With it, I was able to perform calculations on the data and display the contents in HTML. Without PHP, I would be unable to fully create a template with the capability of pulling in information from files and a database. Below is a screenshot of my PHP Database Class:

Taken from db.php within the 'lib' folder.

```php
<?php

    class Database {

        function connect() {
            $db_connect = mysql_connect("localhost", "root", "");
            mysql_select_db("ib_ia") or die(mysql_error());
            return $db_connect;
        }

        function execute($command) {
            if(!mysql_query($command)) return false;
            else return true;
        }

        function query($command) {
```

I was also able to use PHP to help generate a search-engine optimised (SEO) page. I did this by placing individual page and tutorial tags and descriptions within my database, and then printing them out in each individual page's head: (between the <head> and </head> tags) – taken from index.php

Individual page tags

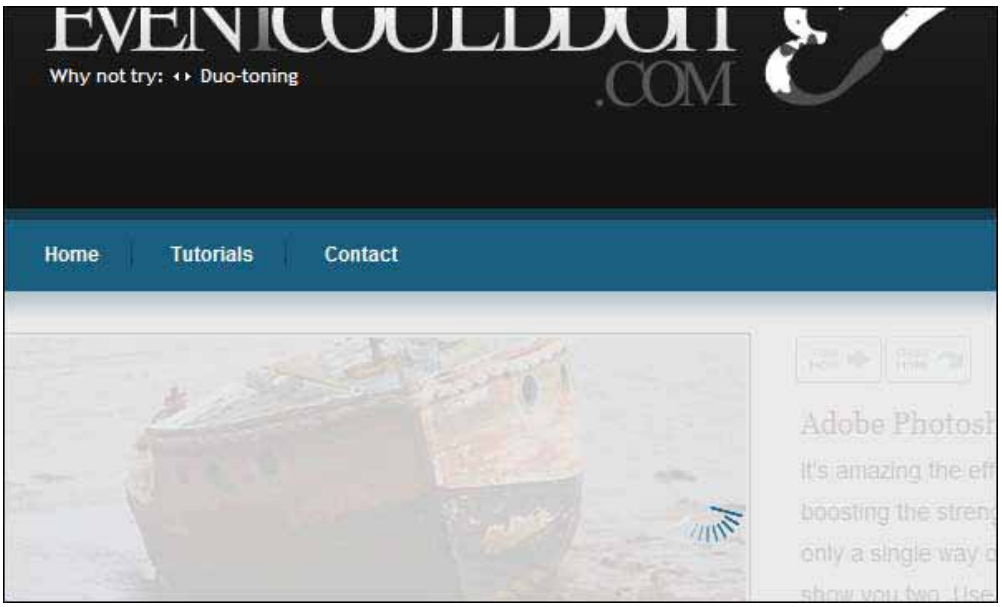
```
&mdash; EvenICouldDoIt.com</title>
"text/html; charset=utf-8" />
ds" content="EvenICouldDoIt web design tutorials <?= $page_handle->page_tags; ?>" />
cription" content="<?= $page_handle->page_description; ?>" />
$cur_theme ?>/stylesheet/default.css" type="text/css" />
```

Being able to insert specific tags into each page promotes a search engines ability to find the website and so aims on maximising my audience (see success criterion 7).

5

Technique 4: Ajax

One of my targets (criterion 8), was to "develop an attractive, intuitive, user friendly interface for the website". For this reason, I decided to link PHP and Javascript to dynamically retrieve a random tutorial for the user to view. If the user didn't show any interest in that particular tutorial, I added a button "show more", that when clicked would retrieve another random tutorial from the database which would fade into the page. A screenshot can be seen below with a loading bar showing whilst the two tutorials change over.

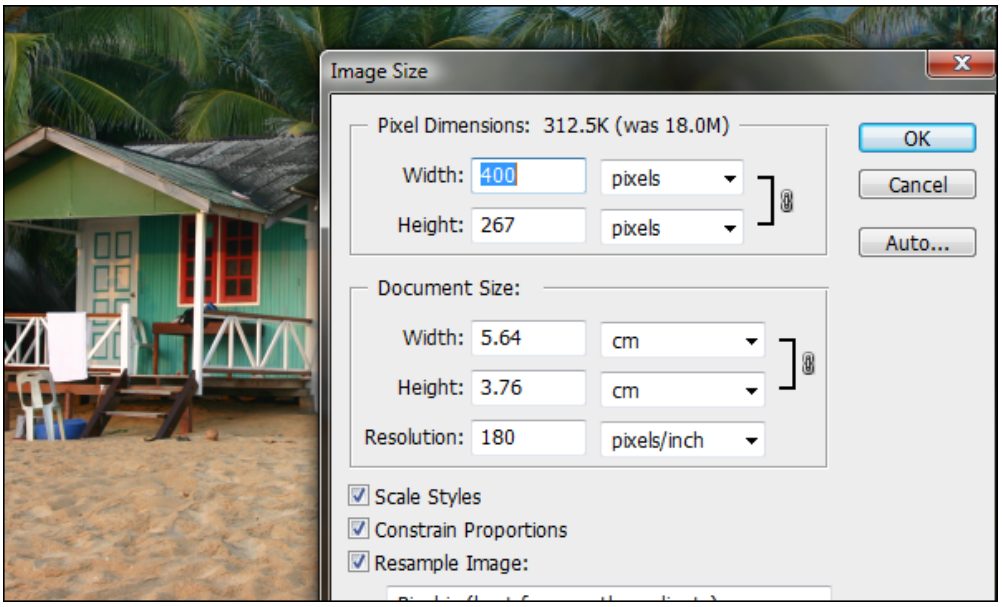


In addition to the large showcase image, a similar "why not try" section at the top of the page (see image above) also aims to promote user interaction with the website. Both uses of Ajax are further effective as they allow the user to be presented with further information without having to load a completely new page, and so load time is reduced.

6

Technique 5: Image Manipulation

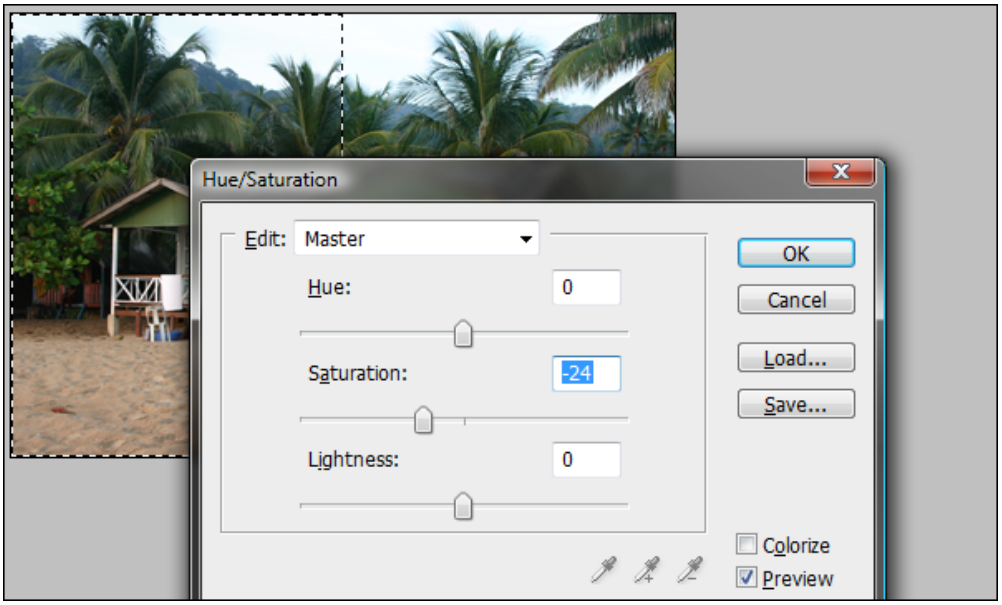
Being a web design tutorials website, images were frequently required. As an example, the 'save for web' tutorial required the use of a photo to demonstrate how images can lose their colour depth. I decided to select a photograph I took whilst on holiday. To implement this in the webpage however, required that I resize the image. Its original size of 3072 x 2048 pixels, and the size of the file were both far too large. Using Adobe Photoshop I was able to resize my image to 400 x 267 pixels (below).



Source: Author's own photograph.

I then added a line and decreased the saturation of one half of the image in order to replicate the effect of this error (below).

7



Source: Author's own photograph.

The final picture can be seen below.

Source: Author's own photograph.

The same technique to resize images was used in every photograph used on the site. Smaller images on webpages not only make the loading time quicker, but also allow the user to see more content at the same time.

8

Other information

Testing for colour-blindness

One simple way to test whether users with colour-blindness can correct view the page is to take a screenshot of the website and turn the image into greyscale. If the website is still coherent in greyscale, colour blind users should be able to interact with it as normal. Below is a screenshot of the website in greyscale.

The image above shows how conversion to greyscale causes no loss in legibility. The website should therefore be accessible by those with colour-blindness.

Words: **941**