Quick & Easy

Web Templates

Build CSS & xHTML 1.0 Compliant Sites,
Design Awesome-Looking Banners,
Create DIV & CSS Based Layouts...
All These Under 60 Minutes!

Quick & Easy Web Templates

LEGAL NOTICE

The Publisher has strived to be as accurate and complete as possible in the creation of this report, notwithstanding the fact that he does not warrant or represent at any time that the contents within are accurate due to the rapidly changing nature of the Internet.

While all attempts have been made to verify information provided in this publication, the Publisher assumes no responsibility for errors, omissions, or contrary interpretation of the subject matter herein. Any perceived slights of specific persons, peoples, or organizations are unintentional.

This book is a common sense guide to building your mailing list. In practical advice books, like anything else in life, there are no guarantees of income made. Readers are cautioned to reply on their own judgment about their individual circumstances to act accordingly.

This book is not intended for use as a source of legal, business, accounting or financial advice. All readers are advised to seek services of competent professionals in legal, business, accounting, and finance field.

This manual is written in Times New Roman for easy reading. You are encouraged to print this book.

Table of Contents

Preface **5**

The Master's Tools **6**

Layouts and Color Schemes **7**

Banner Graphic Design **12**

Tweaking the CSS Code **15**

CSS Typography **18**

In Closing & Some xHTML Reminders **21**

Quick & Easy Web Templates

Preface

Dear Reader,

Welcome and thank you for investing in this powerful manual. This eBook is a focused, topical guide on how to create your own custom-designed website from scratch using tools like Photoshop and freely available applications like Notetab Light.

In this manual, we will take a deeper look into the technical side of things: namely xHTML (eXtensible HyperText Markup Language) and CSS (Cascading Style Sheets). We will explore how to construct complete website layouts using only the very crude and raw Notepad as well as look at the many splendid yet free alternatives out there.

We will also look at ways to create themed graphics and color schemes on your site with a sample case study. So, hop on to the first chapter now!

To Your Quick & Easy Web Templates Success!

The Master's Tools

In this chapter, we shall do a little orientation – discussing what tools and software we have access to. Firstly, I use Adobe Photoshop CS2 for my entire image editing work, but if you have an older version you can definitely follow the steps I use perfectly. If you do not have Adobe Photoshop, a good free alternative will be GIMPShop, an open source image editor that works almost the same as Photoshop.

Download it here:

http://www.softpedia.com/get/Multimedia/Graphic/Graphic-Editors/GIMP.shtml

After that, let me introduce you to Notetab Light. It is a very powerful replacement program for Window's default pure text editor, Notepad. You can download it from http://www.fookes.com/ftp/free/ntfree.zip .

While you're there, you would definitely want to pick up one of their HTML libraries, which is the offline xHTML validator:

http://www.notetab.com/clipbooks/offline-validator.exe

I shall show you what these are for later on, as we come to the coding part.
The last tool you would want to pick up is TopStyle Lite from:

http://www.bradsoft.com/download/index.asp

It's a free CSS Editor that offer a tad more automation for CSS than what plain old Notepad has to offer. With these three software set up properly, we are ready to create our first custom web layout!

Note: You will also need to download the Quicktime Player if you do not have it installed yet. http://www.apple.com/quicktime/download/win.html

Layouts and Color Schemes

During the course of this whole eBook, we shall be learning based on a case study, so let's pretend we'll be building a site for an alarm clock company! Of course, the first step would be to gather materials regarding our subject, which in this case is the alarm clock. I logged on to www.sxc.hu and found a pretty decent picture of an alarm clock – it's included in the workfiles directory of this eBook package. Conveniently, the alarm clock is against a white background so we do not need to do any processing on it. If it were not against a white background, we would have to extract only the clock image and mask out the rest of the background.

Now, as you can see from the image, I have purposely chosen an image that has very limited hues. The dominant colors are yellow, black and gray/silver. That would be very useful in determining our color scheme for the whole layout. Meanwhile, let's take a look at the type of layout we'll be working on:

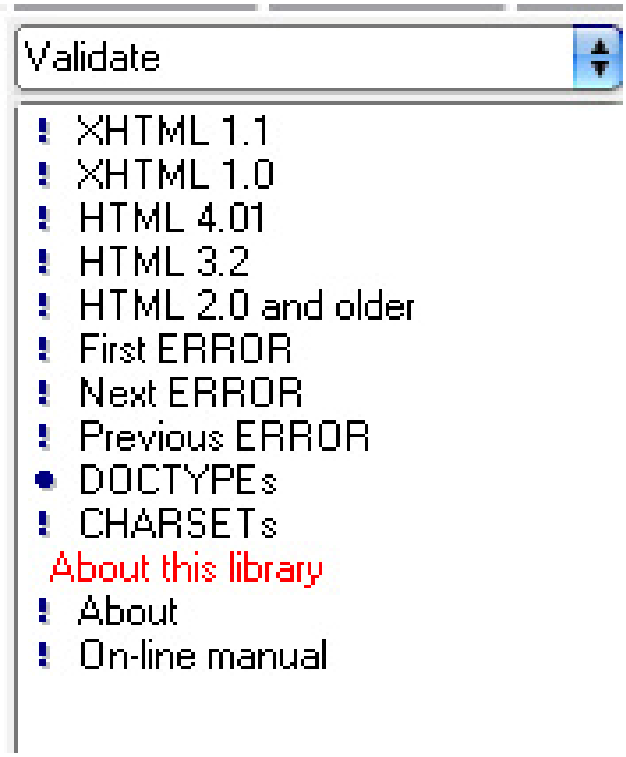
As you can see, it's the traditional layout with a header banner spanning both columns from above, a sidebar on the left and the content on the right.

- Content
- Header Banner
- Sidebar

Now, we're going to create our layout using div tags and CSS as opposed to using the traditional tables method. The reason is very simple: by using clever div tags and CSS code, we are saving up on lines and lines of unnecessary HTML code for tables, hence making the page as small as possible. This equals faster load time for your visitors, especially those with dial up connections.

Now, assuming you've installed Notetab Light and the offline xHTML validator library, open Notetab and click on the “Validate” tab. You will see something like this on the left hand side:

Create a blank document and double click on DOCTYPEs. Then, select xHTML Transitional 1.0 as your document type. You will see something like this appearing on your document:




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

What I would like you to do is delete the line <?xml version="1.0"?> from your code. This tag is not fully supported by most browsers yet, so removing it will ensure that your layout will be displayed consistently across all browsers. The remaining lines of code simply tells the browser that the document's type is an xHTML file. Below that, add a <head> tag to mark the start of your document.

After that, double-click on the CHARSETs item in the library sidebar and choose UTF-8 if your page is purely English. You can add in as many meta tags as you need for the keywords, description, robot indexing etc by clicking on the HTML clip and double-clicking Meta in the sidebar. Now, before you close your <head> tag, there is one final line of code that I would like you to add in:

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

This creates a link between your current page and a stylesheet file called style.css, and anything that is on that file will affect the appearance of your webpage. After that, add a </head> to close the head of your page and start the body (the actual content) with a <body> tag. Now, your code should look similar to this:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Alarm Clock Inc</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-
8" />
<meta http-equiv="content-type" name="description" content="An
Alarm Clock company." />
<link rel="stylesheet" type="text/css" href="style.css" />
</head>
<body>
```

Now, we shall add in the code for our actual layout. First, we need to define the header:

```
<div id="header"></div>
<br style="clear: both" />
```

(The br tag inserts a line break and escapes the sidebar and content DIVs down to the next line)

Then, we shall define the sidebar and content area:

```
<div id="sidebar">This is the sidebar</div>
<div id="content">This is the content area</div>
```

After that, we end our document with </body> and </html>. Then, save the document as a .html file. Next, we shall move on to the stylesheet itself. Fire up Topstyle Lite and you should see a blank document ready. Type the following code into the document:

```
body {
background: yellow;
}
```

Then save it as style.css in the same folder as your .html document.

The line in the CSS file simply tells the browser that the background color of the page is yellow. Straightforward, right? To verify if everything is working OK, just open your page in your browser and you should see that it has a yellow background. If it works correctly, change the background color back to white, because we're only making it yellow for testing purposes.

So, your code should look like this:

```
body {
background: white;
}
```

Banner Graphic Design

Now, let's go to Photoshop for the moment to create the banner for your website. For the banner, we want the image of the alarm clock on the top left of the banner with a reflection to create a classy feeling. Then, we'll add in the title and description big, bold letters to create a neat minimalistic feel, something like what you might see on Apple Computer's site, www.apple.com .

Open up Photoshop (or your image editing application) and load the alarm-clock.jpg file into it. Press CTRL+A to select the whole document and press CTRL+C to copy the image. Then, create a new image by going to File > New. Enter 780 pixels for the width and 150 pixels for the height. Then, press CTRL+V to paste the image of the alarm clock inside it.

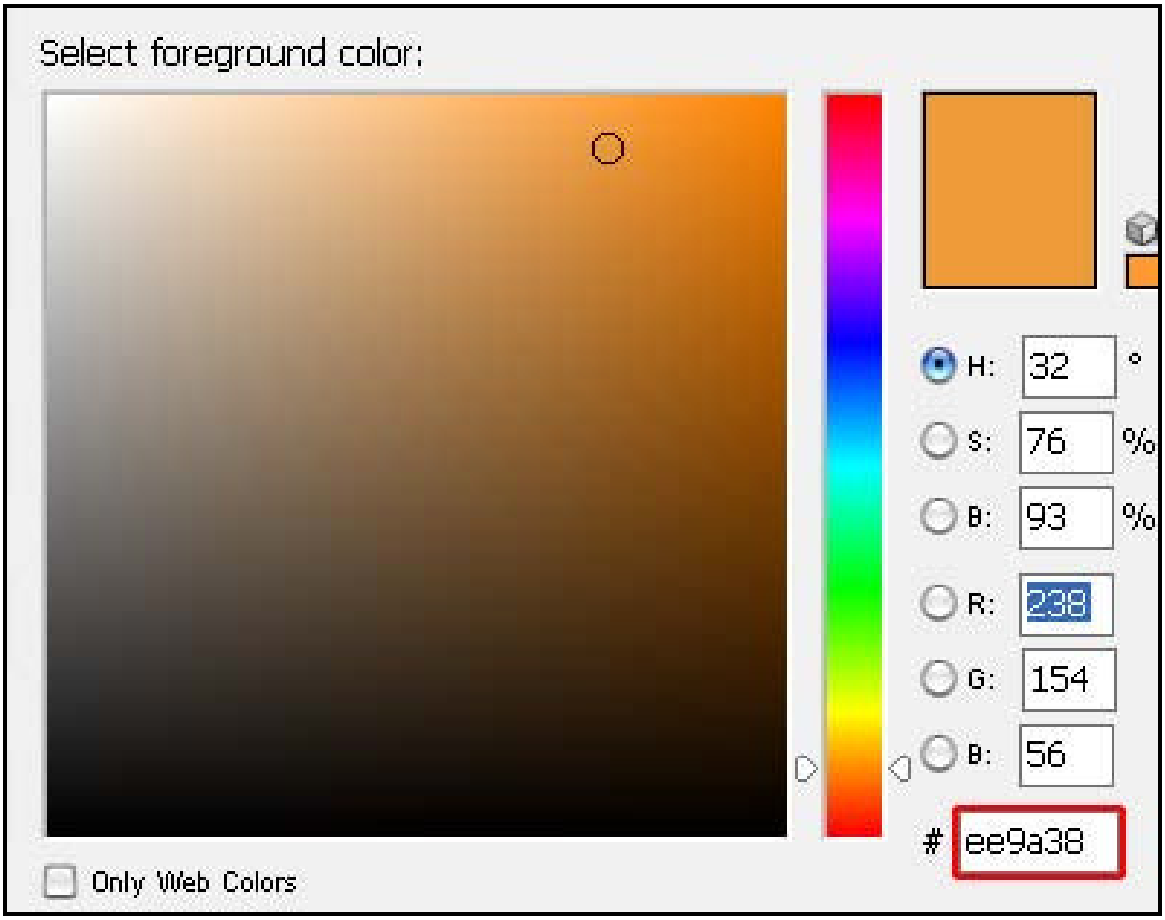
Obviously, the alarm clock image will be too large for the canvas, so press CTRL+T to activate the Transform Gizmo (a bounding box with 8 vertices on the sides) and hold SHIFT while dragging the vertices to fit the image into the canvas. Then, position it to the top left of the whole canvas and press Enter to confirm your transformation. You can see a short video demonstration of it in the transform.mov file.

Then, drag the clock's layer onto the new layer icon to duplicate it and press CTRL+T again for the Transform Gizmo. Hold ALT and drag the top center vertex down to flip it vertically, and position the legs of the now flipped clock to meet the legs of the upright one. Change the layer's blending mode to Multiply.

Next, add a layer mask for the inverted layer. Select the Gradient Tool, press D to set your palette to the default black and white and right click anywhere on the canvas to select the Foreground to Transparent gradient. Finally, drag from the bottom of the banner to the foot of the alarm clock. Again, please refer to the video reflection.mov for a clearer demonstration.

Now, we'll work on the title and description. First, I used the same font for both the title and description for consistency. I used the font Lucida Grande and it can be downloaded

at http://www.jonmega.com/iceman/stuff/fonts/ . Alternatively, you can use the font Tahoma which is installed by default on most Windows based OSes.



Anyway, I switched to the Brush Tool and held the Alt key while dragging my cursor across the alarm clock face to sample the color. When you're using the Brush Tool and you hold the Alt key, that is equivalent to activating the Eyedropper Tool temporarily. I sampled the color #ee9a38 which is a dark orange shade because it works great against our white background. If we used a light shade instead, it would be very hard to distinguish against the white background.

So, using that color, I created the main title for the banner, which is Alarm Clock Inc if you look at the sample files provided. I also used black for the description/tagline and downsized it from the main title's size. Here is the final banner I came up with.


Alarm Clock Inc
Rudely shocking people awake since 1978

Tweaking the CSS Code

Now, we need to pull that banner into your page. Insert the following line of code between your and :

```
<img src="banner.gif" width="760" height="150" alt="Site Banner"
/>
```

The code is pretty straightforward, it defines the source file, width, height and description text for the banner. Now, we need to go to the CSS file and add in the following lines:

```
#header {
width: 760px;
height: 150px;
}
```

Since the #header corresponds with the div with the ID “header”, every property you set in the CSS file within the curly brackets of that #header will be applied to that particular div. In this case, we are just setting the width and height of that div to correspond to the banner image's size so that everything wraps perfectly. Your whole CSS file should now look like this:

```
body {
background: white;
}

#header {
width: 760px;
height: 150px;
}
```

At this stage, the banner should be above the text and the two lines of text will be arranged one on top of the other. Our job now is to make the sidebar go to the left and the

content area to the right. First we need to address the sidebar DIV from our CSS file. To do that, we will use a #sidebar followed by curly brackets:

```
#sidebar {
}
```

First, we'll need to give it a width. I made it 180px wide, plus 10px of margin around the whole box. Hence, the total width of the sidebar DIV would be
180px + 10px (margin on the left) + 10px (margin on the right) = 200px

That leaves 760px – 200px = 560px for the total width for our content area, but let's keep that in mind for the moment. We'll also need to tell the browser that the sidebar DIV goes to the left of the content area instead of above it, so we need to assign a float: left; to it as well. Hence, the overall CSS code for the sidebar will look like this:

```
#sidebar {
width: 180px;
margin: 10px;
float: left;
}
```

Then, we need to deal with the content DIV. First, we assign the same 10px of margin around the box using margin: 10px;. Then, we need to push the content DIV 200px to the right of the page since the first 200px are occupied by the sidebar. However, take note that we have to consider the 10px margin on the left of the content area as well.

The simple boxes above illustrate my point. The blue areas represent the 10px margin, and the white boxes within represent the actual width of the DIVs. If you measure from the left side of the whole thing to the left side of the content DIV, the length will total 210px. Hence, we will need to apply 210px of margin to the left of the content DIV. Last of all, we need to give it a definite width, and its width is equal to 560px – 20px (margins on both sides) = 540px. So, the final CSS code for the content DIV would be:

```
#content {
width: 540px;
margin: 10px;
margin-left: 210px;
}
```

Note that the line margin-left: 210px; should come after the line margin: 10px;. If margin: 10px; were to come after margin-left: 210px;, it would override the code and make all four margins (top, bottom, left and right) 10px, but remember that we wanted 210px for our left margin, right? Remember this: any line in CSS will be overridden by lines affecting coinciding properties after it.

At this point, save your CSS file and refresh your web page. Everything should work fine and all you need to do is to add content into the appropriate boxes. At this point, I usually test my layouts with loads of random text called Lorem Ipsum, which you can find at http://www.lipsum.org/ . In the next section, we shall take a deep look at typography with CSS.

- Sidebar
- Content

CSS Typography

If you've even dabbled in HTML in the past, you might have been taught to use <font> tags to style your fonts, whether it's the color, font family, size and so on. For each paragraphs that you want to apply the same style to, you have to retype or copy/paste the font tags, making it a totally tedious and mundane job. CSS will change all that. Simple as that.

Now, to render our website professional and consistent, we need to have the same style for all headings, another style for all paragraphs, another for all links... Get the idea? If you apply different styles to different blocks of paragraphs in an attempt to “spice up” your boring page, it will definitely backfire and make your page seem that it was done by an amateur.

Nonetheless, let's get to work. First, we open up our CSS file again and look at the body part:

```
body {
background: white;
}
```

So, by adding any text properties in the body section, we will actually be manipulating every element of any text block on our page. That would be useful to define the font family, base size and color of the normal body text. (i.e. Content text blocks)

First, let us start with the base size of the font. There are many methods for manipulating font sizes in CSS, but I shall teach you my favorite (and in my opinion the most efficient) method, which is to set a basic font-size and defining everything else relative to it. To do that, we have to set a base font size by inserting the following line:

```
font-size: 76%;
```

There's a reason why I chose the specific value of 76%, and that's because it simply works. The font size 76% will display consistently across Internet Explorer, Mozilla

Firefox and Opera (and most other browsers), so using that value will ensure your site displays correctly on the majority of your visitor's monitors.

At this point, you might ask what if I wanted different sizes for my headings? Since the 76% is the base font and everything else refers back to it, your heading will still be larger than your paragraphs; it's just 76% of its original size, when you haven't applied the line of CSS. Of course, the paragraph text blocks (and just about everything else) size down too.

We'll also need to set a default font family for the whole page so that in the case that you accidentally leave something unstyled, it will default to that specific font. In fact, you can purposely leave out lines of code if you want them to have the same font as the default, so that saves more space. We do that by adding this line in the body section:

```
font-family: Tahoma, sans-serif;
```

The sans-serif appended behind the font name ensures that if your visitor's computer does not have the font you define (in this case Tahoma), it will choose another font of the same family (i.e. Sans-serif in this case). Sans-serif simply means those fonts that do not have curly ends, for example Tahoma, Trebuchet MS, Verdana, etc.

Finally, we want to set the default color of the text to a dark yellow (almost black) to reduce the great contrast between it and the white background.

```
color: #32312d;
```

So, your final code for the body section should look like this:

```
body {
background: white;
font-size: 76%;
font-family: Tahoma, sans-serif;
color: #32312d;
}
```

Save it and refresh your page to see the changes made.

Next, we'll style our headings:

Let's say we want a different color and font for the headings:

```
h1 {
font-family: Trebuchet MS, sans-serif;
color: #ee9a38;
font-size: 2.0em;
}
```

The code is pretty self explanatory except for the font-size: 2.0em; part. The em unit simply means the height of the letter “m” of the base font size. So, since our regular text is 1em (because we did nothing to change it), our h1 headings are exactly 2 times as tall as our regular text.

You can also do the same for <h2>, <h3>, <ul> (unordered list), <ol> (ordered list) and <a> (link) tags as well. However, I want to show you another example, which is the <a> tag:

```
a:link, a:visited {
color: #ee9a38;
text-decoration: underline;
}

a:hover, a:active {
color: #b16206;
text-decoration: none;
}
```

Notice that it is split into four components, a:link, a:visited, a:hover and a:active. The “link” component is the normal link, “visited” stands for visited links, “hover” is the link when your mouse hovers over it and “active” the state when your mouse button is held down over the link. You can style each component separately, but I styled them like this to simplify the link to normal state and hover state. First, I added this line of code in the .html page:

```
<a href="#">Sample link</a>.
```

Then, I saved both the CSS and HTML files and refreshed the page to see how it works.

In Closing & Some xHTML Reminders

In xHTML, some old, obselete tags such as <font> will not validate correctly. It is important to use more sensible and efficient tags such as and . Also, using DIVs to structure a page involves less work and less clutter. Try comparing that with the amount of <tr> and <td> tags you would need if you made a layout using tables.

One xHTML rule is that every tag should have a closing tag. For example, a should always have a corresponding . For tags that do not have closing tags such as <img> and
, use a self-closing slash to avoid the page not validating correctly. Here's how:

```
<img src="filename" width="value" height="value"
alt="description text" />
```

At the end of the day, a page that does not validate correctly might not matter to you because the page's role is not to validate; it's to present information. However, as technology advances and web pages start to be accessible on cellphones and portable devices, browsers for those devices will adhere to xHTML standards and your page will simply fall apart into an ugly mess if they were loaded on such devices.

Hence, it's very important to observe the simple rules of xHTML on the pages you create. If you are interested in learning more or simply want to find out solutions for problems you may encounter while using xHTML, visit http://www.w3schools.com/xhtml for free xHTML reference. That's where I first started to learn xHTML too!

Having said that, all the best in designing your professional web templates – the quick & easy way!