

Creating websites by hand

By Christian Paratschek - Posted on 2004-11-10 20:17:28
at OSNews [<http://www.osnews.com/>]

With the recent browser statistics, that show Internet Explorer 6, Mozilla/Firefox, Safari/Konqueror and Opera combined at around 95%, it is finally feasible to write modern, CSS-based websites. For many years, this was not possible due to the vast number of legacy browsers, Internet Explorer 4 and 5 and Netscape 4 deployed on the computers around the planet. But with these browsers vanishing, we can finally start to ignore them.

And it's not just the declining number of these browsers. I recently talked to two people that still surf the web with these old browsers, and they both basically said the same: "We're used to having problems with websites, our browsers are old. No problem, that will simply go away when we buy a new computer." While this is of course not representative, it supports my growing opinion that we can start to ignore old browsers. The tipping point for me was that Gecko-based browsers have overtaken IE4/5 and Netscape 4 recently (this is probably a safe bet according to recent web statistics). So it's already more important to have your website work with Mozilla than with any old browser. Basically, if you code standards-compliant HTML and CSS, your site will look the same in Mozilla, Opera and Safari. And while Internet Explorer 6 does have certain problems with stylesheets, it certainly is "good enough", if you know its quirks.

So, today, I want to talk about various aspects of modern webdesign. This may be helpful for all of you who want to write HTML by hand and not use WYSIWYG-tools like Dreamweaver and Frontpage. In case you don't know: WYSIWYG stands for "what you see is what you get". With WYSIWYG-tools you work on your actual website, the program writes the code itself while you put images on the site, create text, and so forth. You may ask: "What is the advantage of writing code by hand?" Well, in my opinion, there are several differences. Dreamweaver MX 2004 actually produces far better code than his predecessors, but you simply can't compare designing with a WYSIWYG-tool to writing code by hand. The latter gives you much more control over the design of your page. Basically, you know, at any time of the design process what you have done, what you are doing and what the consequences of your actions are. "Debugging" your website is a lot easier that way. The code you produce is a lot smaller, thus more efficient. All webdesigners out there will have to deal with HTML and CSS at some time. So it's better if you are prepared! Also, Dreamweaver costs money and may not exist for your platform.

Anyway, designing websites with Dreamweaver is not the topic of this article. All you need today is a text-editor and a browser. Preferably you have more than one browser, at best you can test your website constantly with Internet Explorer 6, Mozilla, Opera and Konqueror. I'd say: at least IE6 and Mozilla are required. Also, a good book about HTML and CSS helps a lot.

Peter Anderson's recommendations:

Castro, Elizabeth. HTML for the World Wide Web (5th Edition). Berkeley: Peachpit Press, 2003.

Cranford Teague, Jason. DHTML and CSS for the World Wide Web (2nd Edition). Berkeley: Peachpit Press, 2001.

Wium Lie, Hakon and Bert Bos. Cascading Style Sheets - Designing for the Web. London: Addison-Wesley, 1999.

Cederholm, Dan. Web Standards Solutions - The markup and style handbook. Berkeley: friendsofED, 2004.

So, first of all, let's explain what we are talking about: HTML is the Hypertext Markup Language. Most of the modern websites are written with HTML (an alternative way to create a website would be Macromedia Flash). CSS stands for "Cascading Stylesheets". HTML is meant to contain all your (preferably well structured) content, stylesheets are there to control the design of your website. "Rendering" is the technical term for what your browser does when it displays a website.

Well, let's get started: open a text editor (preferably one that can highlight syntax, like TextPad (shareware at www.textpad.com), Editeur (shareware at www.studioware.com/), NoteTab Lite (freeware at www.notetab.com), or PSpad (freeware at www.pspad.com/en/index.html) - but Windows Notepad will do). The first thing you will want to do is put in this text:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
  <title></title>
  <meta http-equiv="content-type"
    content="text/html; charset=iso-8859-1" />
  <meta http-equiv="Content-Style-Type" content="text/css"
    />
  <link rel="stylesheet" type="text/css"
    href="link/to/your/stylesheets.css" />
</head>
<body>

</body>
</html>
```

Wow, that was a piece. What's that? Well, the first line is the "Doctype" definition. What we do here is tell the browser that it is about to display an XHTML 1.1 file, this is the most recent HTML standard. If we do this, the browser will expect perfect, well-formed XHTML 1.1, thus rendering our site in standards-compliant (strict) mode, so the results will be a lot more predictable. If we don't include this Doctype, the browser will render the page in "compatibility"-mode, that means it expects old, "bad" HTML and will try to guess what to do with it. Unfortunately, this is exactly what browsers have to do most of the time nowadays. After the Doctype definition, we start out HTML-file with the opening "html"-tag. After this, there's a "head"-section and a "body". Basically, HTML and all Markup

languages work the same way: you open a "tag", with <tag>, then do something, then close it with </tag>. You "mark up" what is between this beginning and closing tag. The head contains some information about the document, the body contains the information that is displayed on the screen. Enter some information into the "title"-tag. This will be what you can read at the top of your browser if you open your website. The following "meta"-tags are not important right now, just be sure to include them. The "link"-tag is interesting. Here you will link to your stylesheet (that still does not exist right now) to make sure that your HTML-file finds the stylesheet. Afterwards, the "body" starts. For now, just write anything into the body (like "Hello, World!"), save your document as "index.html" and open it with your browser.

```
*** Peter Anderson's default blank HTML page ***
*** This page uses the HTML 4.01 Transitional ***
*** standard which is not as strict as the ***
*** newer XHTML standard ***
```

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd"
    ">
<!--      Page created by: Peter Anderson      -->
<!--      Author's e-mail address: pjafrombbay@yahoo.com.au
-->
<!--      Last updated DD/MM/YYYY      -->
<html>
<head>
<title></title>
<meta name="generator" content="TextPad 4">
<meta name="author" content="Peter Anderson">
<meta name="copyright" content="&copy; 2004 Peter Anderson">
<meta name="keywords" content="?, ?">
<meta name="description" content="page description">
<!-- Optional linked stylesheet - change to suite your needs
<link rel="stylesheet" type="text/css" href="C:\Documents and
Settings\Peter Anderson\My Documents\dev\html and
css\mystyles.css">
-->
</head>

<body>

</body>
</html>
```

```
***
```

At first, we will talk about the HTML part. Before you start with your website, you should plan what you want to do. Most of the sites are happy with 2-3 sections. A classic layout will include a header, a menu and the content. You can view a simple example for this on my site [<http://www.homechris.net>]. A tip: view the source code of pages you like. Sometimes you can get really useful information out of it.

There are not too many HTML commands. Here's a short list of the most important tags: `div`, `p`, `br`, `a`, `h1`, (`h2-h6`), `img`, `ul`, `ol`, `li`. I don't even include the dreaded "table", that was used for old-school webdesigns, because i personally rarely ever use it. Probably the first decision you have to make is how to divide your website. Use the "div"-tag to make some basic divisions. The "body"-section of your document could now look like this:

```
<div id="header">
  <h1>Welcome to my Website</h1>
</div>
<div id="menu"></div>
<div id="content">
  <p>Here goes all my content, but I don't have any yet.</p>
  <p>Let's make another paragraph.</p>
</div>
<div id="footer">
  <p>Last updated on ddd mmm yyyy - &copy; Peter Anderson,
    2005</p>
</div>
```

The "id" is an identifier. We need it so that our divisions make sense later on. You can apply this "id" to ANY tag. The stylesheet will use this identifier to assign styles to the document. But we'll talk about this later. So, now we have a website with three sections. We use "h1" (heading nr.1) to write a fat "Welcome"-message into the header-division of our file. No stuff in the "menu"-div, because we don't know yet what menu items to include. In the content, we use the "paragraph"-tag "p" to write some text. Just to make sure, I will emphasize on this: you have to close EVERY tag that you opened, and you have to close it IN THE CORRECT ORDER. Let's talk about the tags I mentioned: we already know that "div" does nothing but define a certain section of your document, "p" creates a paragraph, "h1" (and his smaller brothers "h2" through "h6") are used for headings. "br" creates a simple line break, it's the "smaller brother" of "p". "ul" stands for "unordered list", "ol" for "ordered list". If you use "ol", your browser will automatically add numbers in front of the items. They both need "li" ("list item") to mark up the specific items. Lists look like this:

```
<ul>
  <li>item</li>
  <li>item</li>
  <li>item</li>
</ul>
```

Possibly good for our menu, eh? Now we only have "a" and "img" left: "a" ("anchor") is used for hyperlinks. Use it like this:

```
<a href="http://www.somepage.com">text about somepage</a>
```

Again, the only thing that is actually displayed in the browser is what stands between the tags. In the opening tag, you tell the browser where the link actually goes to. Quite simple, isn't it? You will need "a" tags to "weave" your websites together: linking to another document in the same folder would work this way:

```
<a href="dog.html">View more information about my dog</a>
```

That doesn't restrain to websites, you can also create simple mailto-links, by using

```
<a href="mailto:someone@somewhere.com">mail me!</a>
```

Be careful, these links only work if your viewer has a mailprogram that the browser can call! Now the only thing that's left is "img" (obviously for "image"): we have already learned that HTML is a markup language. So, we can't simply put a picture into our HTML-file. We'll have to put the picture somewhere and tell the file where it is. The according syntax would be:

```

```

Explanation: "src" ("source") tells the browser where the picture is, the information in "alt" is displayed when the picture can't be accessed, or by text-only browsers. One word about the trailing slash: XHTML 1.1 requires you to close ALL tags, even those that are standalone-tags like "img". So, you'll have to write ``, but you can do this a bit shorter by using the above syntax - space and slash. The only other item of our list that needs this special syntax is "br", which should always be written `
`.

So, now, we know the basic ingredients of a HTML file. I encourage you to view the source code of my website, it will be much clearer now. I used "dl" and "dt" ("definition list" and "definition title") instead of "ul", but that's a matter of choice. I added a comment into the body so I don't forget how things work. It's generally a good idea to use comments. They are added within the "body"-section by using `<!--comment-->`. Pretty arcane, eh? Well, once you've remembered that, it's easy. The browser will ignore everything between the opening `<!--` and the closing `-->` when rendering the page. On another note, as you can see in the end of my code, it's easily possible to use images as links. The syntax should be logical to you (if this was a good article until now):

```
<a href="http://somewhere.com"></a>
```

Now, you have a basic introduction into HTML, but, as a matter of fact, we haven't done any actual designing yet. Let's get it on! First, we go back to the "head" of our file. I have talked about the "link"-tag, that goes to a stylesheet.

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

Now it should be a little clearer what it actually does: it's a standalone-tag, so it needs the special closing that "br" and "img" have. The "href"-part should also be clear now, it works just as in a normal "a"-tag. So, we have to put our stylesheet exactly where this reference tells our HTML-file where to expect it. We also define a type "text/css" (this is just standard).

The big advantage of a stylesheet is that we have a central place to control the design of our website. Your website can be composed of thousands of HTML-files, as long as each of them has this link-tag in the head, you control their design from within a single file. HTML is only here to structure the information. Thus, HTML mostly consists of tags for headings, paragraphs, lists, tables, divisions. The clearer the structure of your document is, the better. Stylesheets on the other hand are there to design all these elements. Now, we'll have a short look how stylesheets are constructed. Stay with me, it's a lot easier than you might guess!

```
/* Stylesheet for my new Website */
body {margin:0; font-family:serif; background-color:red;}
h1 {font-size:24px; text-align:center;}
a {text-decoration:none;}
#header {position:absolute; left:20px; top:20px; width:300px;
height:50px;
border:3px solid black;}

<!-- In-line default stylesheet used by Peter Anderson -->
<style type="text/css">
a {
background-color : transparent;
text-decoration : none;
}
a:link { color: navy; }
a:visited { color: navy; }
a:hover{ color: navy; text-decoration: underline; }
a:active { color: navy; }
body {
background-color: white;
/*
background-image: url(? .gif);
*/
color: black;
font-family: Georgia, "Times New Roman", Times, serif;
font-size: medium;
line-height: 150%;
margin-bottom: 20px;
margin-left: 20px;
margin-right: 20px;
margin-top: 20px;
}
h1, h2, h3, h4, h5, h6 {
color: navy;
font-family: Verdana, Arial, Helvetica, sans-serif;
}
hr {
text-align : center;
width: 75%;
}
.note {
font-family: "Courier New", Courier, monospace;
font-size: small;
}
.footer {
```

```
    color: navy;
    font-family: Verdana, Arial, Helvetica, sans-serif;
    font-size: x-small;
}
</style>
```

Let's clear all this up: first of all, comments in stylesheets are added by `/*` and `*/`. That's different to HTML-files. Sucks, eh? Well, unfortunately, there's nothing I can do about it... The basic syntax of a CSS-entry is `item {attribute; attribute;}`. In the first row, for example, we tell our browser that it should use serif-fonts for the whole "body", that our HTML-file has a red background and that it has no margins (you will want to set that to 0 to be sure that your website starts in the upper left corner, most browsers will set a margin of 10 pixels by default if you don't set it explicitly). The next row defines some styles for our "h1"-headings: the font-size should be 24 pixels and the text should be centered. Then we have a very popular definition for hyperlinks: "text-decoration:none" is used to get rid of the underlines of links. Starting to get it? Well, the last one is a little different: remember the "id=header" we used for our first div? Well, with the # in front, we now refer to exactly that division: we tell it something about its positioning, its width and height. I don't want to go into detail too much because I don't want to overwhelm you with CSS commands. There are not too many of them, but you'll have a far easier time with a good book about CSS on your desk (preferably one you even have read once). Trust me, as soon as you have grasped the idea of CSS (which you probably have right now, because, refreshingly, there really is not much more to it than what I have told you), it's easy to read through a small book and get to know all the commands. Again, there are only a few really important ones: margin (distance from an items border to the next item) and padding (distance of the item towards its own border), border (self-explanatory, I guess...), position, top, left, width, height (to put an item somewhere on the page), font, font-weight, font-size, font-family (for assigning special font properties), color, background, text-align are basic examples.

A word on the "cascading" part of the name: it is, for example, sufficient to tell your CSS something like: `body {font-family:verdana;}`. Your browser will assign the verdana font family for every item on the page, you don't have to repeat this for "h1", "p", "a" or whatever. If you, however, decide that you want arial for your hyperlinks, you just add a `{font-family:arial;}` to your stylesheet. Your page will now still be all verdana, with the exception of the hyperlinks, which will be in arial.

In the end of this article I want to illustrate what is possible with stylesheets on the basis of an example that I used for my website. Let's look at the code first. In the HTML-file, we merely have:

```
<div id="header"><h1>/home/chris</h1></div>
```

In the Stylesheet, we have:

```
#header {position:fixed; top:0px; left:0px; height:136px;
width:100%;
    background:url("background.jpg") no-repeat;
    border-bottom:4px solid #a00;}
h1 {font:italic bold 32px serif; text-align:right;
margin:10px 10px 0 0; color:#a00;}
```

Explanation: while I only use two tags in my HTML, a "div" and a "h1", I use the stylesheet extensively to design the header and the heading: first, I assign a position to my header, a height of 136 pixels and a width of 100% (yes, percentages are possible). Also, I define that the header-div should use a jpg as background. "no-repeat" means that this jpg should be displayed just once (you can also use a small picture and repeat that over and over again to get some kind of a muster). Finally, I define a border on the bottom of the div with 4 pixels, solid (versus dotted, dashed and a few other possibilities) and this border should be some kind of red. Then I define for my heading, that it should have an italic, bold, 32 pixels big serif-font. It is text-aligned on the right side, but with a margin of 10 pixels to the top and the right (and zero pixels to the bottom and left, this works clockwise), so that it doesn't "glue" to the upper, right corner of the browser-window. The heading gets the same color as the border of the div, a00. So, with some heavy stylesheet use, I was able to completely separate design and content. If I want to redesign my website today, I do not have to touch any file except for the stylesheet!

Almost done, there are just a few more things that I want to mention at the end of this article. You can use more than one stylesheet at the same time. You can, for example, use a second stylesheet to control the print layout of your site. Just add this to your header:

```
<link rel="stylesheet" title="standard" media="screen"
href="screen.css" type="text/css" />
<link rel="stylesheet" title="print" media="print"
href="print.css" type="text/css" />
```

We just added a "media"-attribute. A browser uses the first stylesheet to display the site on the computer monitor and the second stylesheet on the printer. Both styles may look completely different and can exist completely independently from each other. You can even add special divs (or any other items for that matter) that get the attribute `display:none;`, so they don't get displayed either on the screen or on the printer. The possibilities are endless. A final tip for all webdesigners out there: test your code constantly. Often, when I do something difficult, I just add one attribute at a time and immediately look what changes on screen occur before I change/add another attribute. Probably the biggest advantage over designing with a WYSIWYG-tool is that you retain complete control over the design and creation process. Arcane errors that are difficult to debug almost never happen.

So, that's it for now. Some more eyecandy can be found at csszengarden.com [<http://csszengarden.com/>], another interesting site with great articles is alistapart.com

[<http://alistapart.com/>]. I really hope that some of you could put this to good use. Thanks for your attention!

Copyright OSNews.com 1997-2004. All Rights Reserved. OSNews and the OSNews logo are trademarks of OSNews.
Modified by Peter Anderson (pjafrombbay@yahoo.com.au)