**Basic Concepts: Part 1**

An introduction to domain names, web servers, and website hosting

I assume that you know nothing about the inner workings of the Internet; maybe you’re not even sure how people actually get to web sites, where the web sites are actually sitting, what the web is in the first place….

In this article I am going to give you the minimum you need to get your ‘feet wet’ so that we can quickly get into building web sites. I won’t go into painful micro-details that would put all but true nerds to sleep, again there is just enough so that you have a basic understanding of what’s going on.

**What is the web?**

In a nutshell, the web is a whole bunch of interconnected computers talking to one another. The computers (on the web) are typically connected by phone lines, digital satellite signals, cables, and other types of data-transfer mechanisms. A ‘data-transfer mechanism’ is a nerd’s way of saying: a way to move information from point A to point B to point C and so on.

The computers that make up the web can be connected all the time (24/7), or they can be connected only periodically. The computers that are connected all the time are typically called a ‘server’. Servers are computers just like the one you’re using now to read this article, with one major difference, they have a special software installed called ‘server’ software.

**What is the function of server software / programs?**

Server software is created to ‘serve’ web pages and web sites. Basically, the server computer has a bunch of web sites loaded on it and it just waits for people (via web browsers) to request or ask for a particular page. When the browser requests a page the server sends it out.

**How does the web surfer find a web site?**

The short answer is: by typing in the URL, or in other words, the web site address. So for example, if you wanted to find the web site www.killersites.com, you would type in the address into your web browser’s address bar or maybe use your ‘favorites’ or ‘bookmarks’ link to Killersites.

There are other ways to find web sites (like search engines,) but behind the scenes web sites are all being found by going to the web site’s official address. That brings us our last nerd detail: how does a website get an official address so that the rest of the web can find it?

**Registering your domain name**

If you ever wondered what the heck registering a domain was all about … you probably figured it out by now! But just in case – registering a domain name gets you an official address for your web site on the World Wide Web. With this ‘official’ address, the rest of the web can find you.

Like your home address is unique in the real world, there also can’t be any duplicate addresses on the Internet, otherwise no one would know where to go! In other words, domain names are unique addresses on the web.

**Why does registering a domain name cost money?**

If you want to have your own unique address on the web, your own domain name, it will cost a few bucks for each year you want to ‘own’ the name. The cost of registering a domain name ranges from less than $10 USD to about $30 USD per year. You can register a domain from 1 to 10 years.

The reason for the cost is that the central ‘address book’ of all the world’s domain names needs to be updated – somebody’s got to pay for that! You may have noticed that I just snuck in a little extra piece of information: the giant ‘web address book’ of domains.

That leads us to our last bit of nerd information: when you type in a website’s domain name or click on a link that takes you to that domain name, your browser starts asking servers where that particular domain name is sitting (on the web) and the servers are then able to tell the browser where to go by referring to the giant address book I mentioned above.

**Building your first web page: Part 1**

In this ‘hands-on’ module we will be building our first web page in no time. We just need to quickly cover a couple of points beforehand to help get our feet on the ground.

The three ways you can build a web page

1. **Use a pre-made template: WHAT IS A WEB DESIGN TEMPLATE?**

A web site design template is a pre-made website design template which can be customized to reflect your company’s branding. Website design templates can be found in various formats like Photoshop and HTML. Many times, these templates are compatible with HTML editors like GoLive, FrontPage, and Dreamweaver.

Web site templates can be very useful; they can be used by experienced web designers to ‘jump-start’ the creation of a website. They are also a way for people to put out great-looking web sites quickly with little or no knowledge of HTML and web design.

**2. Use an HTML editor like FrontPage or Dreamweaver:**

HTML editors make building web pages feel like (to a certain extent) creating a document in Microsoft Word … it’s made pretty easy. But the downside is that you lose a certain amount of control of what you’re doing and in some cases become dependent on the program.

**3. Hand-code your HTML in a text editor like Notepad:**

That means you type in the HTML code yourself. This is the approach we are going to use here, because it’s the quickest way to learn how to build web pages, and it is arguably the best way because you have the most control over what you’re doing.

Ok, now that we know the advantages of hand-coding web pages, let’s jump into just the bare minimum of theory, and then we will build our first web page!

**What are HTML tags?**

HTML tags are specifically formatted text that creates ‘markers’ for web browser to read and interpret. These ‘markers’ tell the web browser what and how to display things on the web page. Tags are placed in and around text and images (text and images are some of the ‘things’) that you want to have appear in your web pages.

HTML has a whole bunch of tags (just like the alphabet has a whole bunch of letters) that the web designer can use to build web pages. As mentioned above, tags have a specific structure so that when the browser is reading an HTML page, it knows the tags from the normal text.

Tags are typically words or abbreviations of words placed between angled brackets. So for example: to make text bold, HTML has the ‘bold’ tag that looks like this:

<b>This text will be bolded</b>

Another commonly used tag is the paragraph tag:

<p>

This is a paragraph of text.

</p>

**Building a web site: Part 1**

Now that you’ve created your first web page and have some of the basics down, we can move onto building a web site.

**Introduction: What is a website?**

A website is just a bunch of web pages connected together through something called links. In HTML there is a special tag called (you guessed it!) the ‘link’ tag, and it looks like this in its most basic form:

<a href="..."></a>

And here is an example of the link tag with a destination filled in:

<a href=" http://www.killersites.com ">Go to killersites.com</a>

It’s safe to say that we have all used links when surfing the web. Whenever you click on a link that takes you to another page, you are using a link tag. The link tag is the most important tag in HTML; it makes the Internet interconnected!

In the above example, we see that the link tag points to the web site www.killersites.com and the text that is displayed on the web page is: ‘Go to killersites.com’. So if you wanted to create a link that took someone to www.yahoo.com, for example, you would replace the www.killersites.com with www.yahoo.com, Yahoo’s address.

Like all other tags, link tags have an opening tag (<a href=” http://www.killersites.com “>) and a closing tag (</a>). Links tags are a little more complex than the other tags we have seen so far, but I think that you can handle it!

Some of you may have noticed that there is some text in the link tag that comes before the web site address; the text I am talking about is this:

http://

This text tells the browser that the link is pointing to a web page. Sometimes links can point to other things besides web pages; things like movies, PDF files, and so on.

**Absolute vs. Relative URL**

To link pages in your web site from one page to the next you have a choice of using one of two types of addresses: absolute addresses (complete) and relative addresses (partial).

Before I go on, URL is a nerd’s way of saying ‘address’.

An absolute URL is the complete address of a page that can be found from any other location on the Internet. So let’s say you have a page called contact.html on the root of your web site who’s domain name is www.myStore.com . In this case, the absolute URL of the contact.html page would be:

'http://www.myStore.com/contact.html'

Ok, now I know I lost a few people because I used a word that I did not explain: ‘root’.

When geeks talk about the root of a web site, they are taking about the base of the web site, the starting level.

The files (pages, images, etc.) that make up your web site are organized in folders just like any other files that you store on your home computer. Your host will give you a space/directory on their server for you to place all your website’s files.

This space/folder assigned to you will be the ‘root’ of your web site. This means that as far as the Internet is concerned, anything (html files, images, other folders, etc.) in this folder is directly accessible by your domain name plus the name of the item. Huh! Even I’m a little confused! Perhaps a little example will explain this better:

Let’s say that on the root level of your website, you had these HTML files:

index.html and contact.html

And in a folder called ‘products’, you placed whole bunch of other pages with one called ‘bookcases.html’. You decided to put all your ‘product’ HTML pages into a ‘products’ folder to keep the web site more organized – a smart thing to do!

Our next step is to learn **CSS (Cascading Style Sheets),** a key tool for today’s web designers.

**What is CSS?**

CSS is the acronym for: ‘Cascading Style Sheets’. CSS is the sister language to HTML that allows you to style your web pages. An example of a style change would be to make words bold. In standard HTML you would use the <b> tag like so:

<b>make me bold</b>

This works fine and there is nothing wrong with it per se, except that now if you wanted to, say, change all your text that you initially made bold to underlined, you would have to go to every spot in every page and change the tag.

Another disadvantage can be found in this example: say you wanted to make the above text bold, make the font style Verdana and change its color to red you would need a lot of code wrapped around the text:

<font color="#FF0000" face="Verdana, Arial, Helvetica, sans-serif">

<b>This is text</b></font>

This is verbose (wordy) and contributes to making your HTML messy. With CSS you can create a custom style elsewhere and set all its properties, give it a unique name and then ‘tag’ your HTML to apply these stylistic properties:

<p class="myNewStyle">My CSS styled text</p>

And in between the <head></head> tags at the top of your web page you would insert this CSS code that defines the style we just applied:

<style type="text/css">

<!--

 .myNewStyle {

 font-family: Verdana, Arial, Helvetica, sans-serif;

 font-weight: bold;

 color: #FF0000;

 }

-->

</style>

In the above example we include the style sheet code inline, or in other words, in the page. This is fine for smaller projects or in situations where the styles you’re defining will only be used in a single page.

There are many times though when you will be applying your css styles to many pages and it’s a hassle to have to copy and paste the CSS code into each page. Besides the fact that you will be cluttering up each page with duplicate CSS code, you also find yourself having to edit each of these pages … if you want to make a style change.

The solution: like with JavaScript, you can define/create your CSS style in a separate file and then link it to the page you want to apply the code to:

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

The above line of code links your external style sheet called ‘myFirstStyleSheet.css’ to the HTML document. You place this code in between the <head> </head> tags in your web page.

To create an external style sheet all you need to do is create a simple text document (on Windows you simply right-click and select new -> text document) and then change it from a file type .txt to .css. You can change the file type by just changing the file names extension. The file name’s extension on Windows tells the computer what kind of file it is and allows the computer to determine how to handle the file when, for example, you try to open it.

You probably guessed it; CSS files are just specially (specifically) formatted text files much in the same way that HTML pages are. There is nothing special or different about the file itself, rather it is the contents of the file that makes a CSS document a CSS document.

**When working with an external CSS document there are a couple of points to remember:**

**1.** You DON’T add these tags in the CSS document/page itself as you would if you embedded the CSS code in your HTML:

<style type="text/css"></style>

Since the link in your web page connecting the CSS page to your HTML page says that you are linking to a CSS page, you don’t need to declare that the code in the page is CSS. That is what the above tags do. Instead you would just add your CSS code directly to the page:

.myNewStyle {

 font-family: Verdana, Arial, Helvetica, sans-serif;

 font-weight: bold;

 color: #FF0000;

}

.my2ndNewStyle {

 font-family: Verdana, Arial, Helvetica, sans-serif;

 font-weight: bold;

 color: #FF0000;

}

.my3rdNewStyle {

 font-family: Verdana, Arial, Helvetica, sans-serif;

 font-weight: bold;

 font-size: 12pt;

 color: #FF0000;

}

In the above example I have created a series of CSS classes that can be applied to any HTML tag like so:

<p class="myNewStyle">My CSS styled text</p>

or

<h2 class="my3rdNewStyle">My CSS styled text</h2>

You will notice that in the above example I applied a CSS style to an <h2> tag. This tag sets the size of the text that it wraps to a size that is preset in the browser (ex: 10 pixels). When you apply a CSS class to it, the CSS code overrides the default size that you would normally get with an <h2> tag in favor of the size specified in the CSS class. So now you can see that CSS can override default HTML tag behavior!

In the above examples, I have CSS code where I define my CSS classes and then ‘apply’ them to various elements in the page. Another way to apply CSS is to globally redefine an HTML tag to look a certain way:

h1 { font-family: Garamond, "Times New Roman", serif; font-size: 200%; }

What this CSS code does is set the font style and size of all <h1> tags in one shot. Now you don’t have to apply a CSS class as we did before to any <h1> tag since they are automatically all affected by the CSS style rules.

Here is another example of where I give the whole page bigger margins:

body { margin-left: 15%; margin-right: 15%; }

As you can see, you can redefine any tag and change the way it looks! This can be very powerful:

div {

 background: rgb(204,204,255);

 padding: 0.5em;

 border: 1px solid #000000;

}

Set in the above code, any <div></div> tag will now have a background color of ‘rgb(204,204,255)’ and have a padding of 0.5em and a thin 1 pixel border that is solid black.

**A few things to explain about the above:**

Color in CSS can be expressed in a few ways:

1. In Hex -> for example: #000000 – this is black and this: #FF0000 is red.
2. In rgb -> rgb(204,204,255) is a light purple-blue color.
3. With named colors like: ‘red’ or ‘blue’

I typically use hex color since I am familiar with them or I just use named colors. So the last example can be rewritten like so:

div {

 background: green;

 padding: 0.5em;

 border: 1px solid #FF0000;

}

So instead of ‘rgb(204,204,255)’ , I just specified ‘green’.

By using RGB (RGB is the acronym for: ‘Red Green Blue’) and Hex color, you can really get the exact color you want easily when you know your codes. Luckily many programs (like Dreamweaver) provide easy to use color pickers for you so you don’t need to know the values for the code.

In this last example I will show you the ‘super cool’ CSS code that allows you to create link roll-over effects without images:

:link { color: rgb(0, 0, 153) } /\* for unvisited links \*/

:visited { color: rgb(153, 0, 153) } /\* for visited links \*/

:hover { color: rgb(0, 96, 255) } /\* when mouse is over link \*/

:active { color: rgb(255, 0, 102) } /\* when link is clicked \*/

The above CSS will cause your link to change color when someone hovers their mouse pointer over it, instant rollovers with no images! One important note with the above code is that it is important that the style declarations be in the right order: "link-visited-hover-active", otherwise it may break it in some browsers.

CSS is very powerful and allows you to do things that you can’t do with standard HTML. It is supported nicely now in all the modern browsers and is a must learn tool for web designers.

The above examples are just a small sample of what you can do with CSS, but it should be more than enough for you to start styling your pages nicely. Like with many technologies, CSS has a lot of capability that most people will not need to use often if at all. Don’t get caught in the trap of thinking that if there is some functionality/feature available, that you have to use it.

**CSS Tutorial: Part 1**

Now that we have the first few lessons under our belts, we will be taking the next step towards total nerd glory with this first of a three-part lesson on creating a pure CSS web page.

Not just CSS styling, but CSS positioning too, that means NO TABLES required!

Once completed, you will be creating pages in sophisticated ways that will allow your web sites to be seen on all types of devices including normal computers, cell phones, Iphones, PSPs … any device really!!

… That also means you will be able to put out websites with much less work where your web pages will load faster and will be easy to update and print!

**CSS Template Layout – Part 1 of 3**

In this easy to understand (I hope!) tutorial on creating web pages with CSS, I will concentrate on the key components of css, where (by the end of the tutorials) you will be able to create nice looking CSS based web pages.

After completing this tutorial, you should have enough information to explore CSS and web page design even further … into to areas I don’t cover here.

Let’s get started:

You can download the CSS and HTML files we build in this tutorial: **CSS files**

You can check out what the page should look like here: **CSS Example Page**

**Tags used in this CSS based layout:**

Because of the power of CSS, we will be able to reduce the number of HTML tags we use in a page big time, all the while still being able to lay out great-looking pages using only 6 types (for lack of a better word) of HTML tags.

**The tags we will use to layout the content:**

1. **<h.>** The Heading tags which range from **‘<h1></h1>**’ to **‘<h6></h6>’** , are going to be used to mark/tag headings in our pages. So the most important heading will be wrapped in a **<h1>** tag and the least important in a **<h6>** tag. An example of a heading for this article:

<h1>CSS Template Layout</h1>

This tells the browsers and the search engines, too, that this page is primarily about: ‘CSS Template Layout’

All browsers have a default size (for each **<h.>** tag) as to how it renders text placed between these tags. Many of these defaults can be unusable (especially **<h1>**) because they come out too big. But never fear, CSS is here, and we will use CSS to make the text sizes more to our liking.

2. **<p>** The Paragraph tag is used to mark parts of the pages as being ‘paragraphs’, simple enough. Paragraph tags are what is called a ‘block element’; that means that it acts like a block where a space is automatically inserted before and after each **<p>** tag pair. You will see it work in the examples coming up.

3. **<ul>** and **<ol>** List tags will be used to create our menus. The tag **<ul>** is the ‘un-ordered list tag’ that creates a list with bullets or other images/icons that do not specify or denote an order; hence the term ‘un-ordered’. The other list tag mentioned **(<ol>)** is the ‘ordered list tag’ and it creates a list that, instead of with bullets, marks the list items with numbers or letters. Code examples to follow.

4. **<div>** and **</div>**: Div tags allow you to demark a portion of your page so that you can do things to it. Another way of saying ‘demark a portion’ can be ‘to put into a container’. Once a part of your web page is in this <div> container you can do all kinds of stuff like style it, animate it, make it visible or invisible, and so on. Div’s represent the next generation of formatting HTML pages that, in many ways, are superior to tables.

We will use div’s to create containers for parts of our page. One div will be used to ‘hold’ our navigational menu and another div to ‘hold’ the main page.

5. **<a href>** The most important tag in HTML: the ‘link tag’ or the ‘hyperlink tag’. This makes text ‘hyper’ so that when we click on it we can load another page or activate/call some JavaScript (otherwise known **as ECMA script).**

6. **<img>** This is the ‘image tag’, which allows you to link to images so that they show up in our pages. In HTML, images are not embedded into the actual page, instead the image tag **(<img>)** only points to where the image is and the browser will attempt to load that image when a surfer loads your HTML page.

That covers the HTML tags we will use in our layout! No need for table tags, **<br>** tags, and nasty (and DEPRICIATED) **<font>** tags.

Creating the basic page template

**To work through the examples, we are going to need a practice HTML page.**

**To create the practice HTML page do the following:**

1. Go to your desktop and create a simple text document. On Windows all you need to do is right-click and select: New -> text Document.

This will create a simple blank text document on your desktop. Name the file: practiceHTML.htm. Windows will show you a warning asking you if you want to change the file extension. Just say ‘yes’. Now right click on the file and select: ‘open with’ and select ‘Notepad’. Once inside Notepad, paste in the template code found in the grey box below:

(A simple way to cut/paste text in Windows is to press and hold the Ctrl key and ‘c’ for copy and ‘v’ for paste.

<DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"

"http://www.w3.org/TR/html4/strict.dtd">

<html>

 <head>

 <title>My Practice HTML Page</title>

 <meta http-equiv="Content-Type"

 content="text/html; charset=iso-8859-1">

 </head>

 <body>

 </body>

</html>

This code forms the basic structure for all HTML pages. You can now cut and paste the code samples in between these tags:

<body><!--Insert code here! --></body>

One last note: in between the <body> </body> you find what is called a comment. Comments are a way to place notes that cannot be seen in the browser. Anything in between the following characters becomes an HTML comment:

<!-- and -->

Anything inside comments will be invisible in the browser. So in the above example the word ‘and’ would be invisible in the browser. It is a good way to leave messages about what you are doing in the page. It can come in handy later on when someone else is working on the page or even when you are, because you may (you WILL) forget why you did things a certain way.

Building the CSS

Once you have created the template page, create a folder and name it something like: ‘myCSSwebsite’ and then drop the HTML page into it. In that same folder, create a new text document and call it: ‘myCSS.css’. Once created open that file and paste in this template CSS code and then save it:

/\* Generic Selectors \*/

body {

 font-family: Georgia, &quot;Times New Roman&quot;, Times, serif;

 font-size: 14px;

 color: #333333;

 background-color: #F9F9F9;

}

p {

 width: 80%;

}

li {

 list-style-type: none;

 line-height: 150%;

 list-style-image: url(../images/arrowSmall.gif);

}

h1 {

 font-family: Georgia, &quot;Times New Roman&quot;, Times, serif;

 font-size: 18px;

 font-weight: bold;

 color: #000000;

}

h2 {

 font-family: Georgia, &quot;Times New Roman&quot;, Times, serif;

 font-size: 16px;

 font-weight: bold;

 color: #000000;

 border-bottom: 1px solid #C6EC8C;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Pseudo classes \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

a:link {

 color: #00CC00;

 text-decoration: underline;

 font-weight: bold;

}

li a:link {

 color: #00CC00;

 text-decoration: none;

 font-weight: bold;

}

a:visited {

 color: #00CC00;

 text-decoration: underline;

 font-weight: bold;

}

li a:visited {

 color: #00CC00;

 text-decoration: none;

 font-weight: bold;

}

a:hover {

 color: rgb(0, 96, 255);

 padding-bottom: 5px;

 font-weight: bold;

 text-decoration: underline;

}

li a:hover {

 display: block;

 color: rgb(0, 96, 255);

 padding-bottom: 5px;

 font-weight: bold;

 border-bottom-width: 1px;

 border-bottom-style: solid;

 border-bottom-color: #C6EC8C;

}

a:active {

 color: rgb(255, 0, 102);

 font-weight: bold;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* ID's \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#navigation {

 position: absolute;

 z-index: 10;

 width: 210px;

 height: 600px;

 margin: 0;

 border-right: 1px solid #C6EC8C;

 font-weight: normal;

}

#centerDoc {

 position: absolute;

 z-index: 15;

 padding: 0 0 20px 20px; /\*top right bottom left\*/

 margin-top: 50px;

 margin-left: 235px;

}

Don’t let the CSS freak you out, I will explain the important details and you will soon see how easy it really is. One last thing for you to do before I finish this part of the tutorial, we need to add some code to our HTML page.

In between the <body></body> tags you will need to insert this code:

<div id="navigation">

<h2>The Main navigation</h2>;

</div>

<div id="centerDoc">

<h1>The Main Heading</h1>

<p>

Go to the Web Designer's Killer Handbook home page and grab the practice HTML page that we will used as the starting template for this tutorial. You can find it under the heading:

**'To create the practice HTML page do the following:’**

Follow the instructions there and create your basic HTML page - and do it now!

</p>

</div>

And in between the <head> </head> tags you will need to insert this:

<title>First CSS Tutorial</title>

<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

<link href="myCSS.css&quot; rel=&quot;stylesheet" type="text/css">

With this in place we will be able to start styling our page. If you take a look at the HTML page now you may be surprised to see that we already started!

If you haven’t set the page up yet, please do so to make sure you have everything working thus far. Should you have any problems, go to the KillerSites.com Forum and post your questions.