Definition Lists *(no bullet points or numbers unlike <ol> and <ul>)*

Example:

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<dl>
  <dt>Term 1</dt>
    <dd>Definition of term 1</dd>
  <dt>Term 2</dt>
    <dd>Definition of term 2</dd>
    <dd>Definition of term 3</dd>
</dl>
</body>
</html>
```

Table Tag

The table tag displays information in rows and columns.

I have spent more hours than I care to think about messing with tables. While not hard to understand, it can be a bit tricky to getting everything opening and closing in the right order.

The following code:

```
<table>
 <tr>
  <td>Row 1 - Col 1</td>
  <td>Row 1 - Col 2</td>
 </tr>
 <tr>
  <td>Row 2 - Col 1</td>
  <td>Row 2 - Col 2</td>
 </tr>
</table>
```

Produces a table that looks like:

Row 1 - Col 1 Row 1 - Col 2

Row 2 - Col 1 Row 2 - Col 2

Inside the <table> tag are <tr> or table row tags which in turn contain <td> or table cell tags.

Data goes in the <td> cell. That data can be text, other tags or whatever. Remember that first comes <table>, then <tr>, then <td>, then and only then, data.

Copy the table above and paste it into one of the pages you made before. If you skipped that lesson (tut-tut) take a look at http://www.html-5-tutorial.com/p-tag.htm.

Below I added some CSS and the border attribute (border="1") to better display the rows and columns.

Row 1 - Col 1 Row 1 - Col 2

Row 2 - Col 1 Row 2 - Col 2

Here's the rub. There has to be the same number of cells in each row unless you indicate a table cell spans multiple columns and/or rows.

Multiple columns:

```
<table border="1">
 <tr>
  <td colspan="2">Row 1 - Col 1 & Col 2</td>
 </tr>
 <tr>
  <td>Row 2 - Col 1</td>
  <td>Row 2 - Col 2</td>
 </tr>
</table>
```

Row 1 - Col 1 & Col 2

Row 2 - Col 1 Row 2 - Col 2

Multiple rows:

```
<table border="1">
 <tr>
  <td>Row 1 - Col 1</td>
  <td rowspan="2">Row 1 & Row 2 - Col 2</td>
 </tr>
```

```
 <tr>
  <td>Row 2 - Col 1</td>
 </tr>
</table>
```

Row 1 - Col 1 Row 1 & Row 2 - Col 2

Row 2 - Col 1

Or a combination of both:

```
<table border="1">
 <tr>
  <td>Row 1 - Col 1</td>
  <td>Row 1 - Col 2</td>
  <td>Row 1 - Col 3</td>
  <td rowspan="4">Row 1, 2, 3 & 4 - Col 4</td>
 </tr>
 <tr>
  <td>Row 2 - Col 1</td>
  <td colspan="2" rowspan="3">Row 2, 3 & 4 - Col 2 & 3</td>
 </tr>
 <tr>
  <td>Row 3 - Col 1</td>
 </tr>
 <tr>
  <td>Row 4 - Col 1</td>
 </tr>
 <tr>
  <td colspan="2">Row 5 - Col 1 & 2</td>
  <td colspan="2">Row 5 - Col 3 & 4</td>
 </tr>
 <tr>
```

```
 <td colspan="4">Row 6 - Col 1, 2, 3 & 4</td>
 </tr>
</table>
```

Row 1 - Col 1 Row 1 - Col 2 Row 1 - Col 3 Row 1, 2, 3 & 4 - Col 4

Row 2 - Col 1 Row 2, 3 & 4 - Col 2 & 3

Row 3 - Col 1

Row 4 - Col 1

Row 5 - Col 1 & 2 Row 5 - Col 3 & 4

Row 6 - Col 1, 2, 3 & 4

As I said, tables can get tricky – and I'm not done. In addition to the <tr> and <td> tags there are:

<th> - Table header

<colgroup> - Column Group

<thead> - Table head

<tbody> - Table body

<tfoot> - Table foot

The day may come when you want to make a table with all the bells and whistles so it's good to know they exist, but as a beginner you don't need them.

Also, you should know that in HTML 4.01 tables could have various attributes. The opening table tag looked something like this:

<table cellpadding="5" cellspacing="5" border="1">

In HTML5 most attributes (eg. cellpadding="5") have been eliminated. Use CSS instead. This of course assumes you know CSS (which you probably don't) but take my word for it, web design is better off without them. I'll introduce CSS in the div tag page.

As I said above, getting tables right can be a pain in the neck. One attribute that was going to be eliminated but wasn't is the border attribute. I often add the border attribute (<table border="1">) to get the table debugged and then remove it (<table>) when ready to publish. Normally I use CSS to add borders to tables as I don't like the look of the border that the border attribute gives, but I use them on this page so you can see it work.

Take some time and play with tables. Copy and paste the tables above into your webpage, then edit them and see what happens.

As web pages are usually designed in columns and rows it's tempting to use tables for a page's layout. Don't – that's not their purpose.

Spiders don't always read tables well so they can negatively affect SEO.

Tables need to be completely downloaded before they are displayed and that can leave visitors staring at a blank screen – the kiss of death.

A big complicated table can be a nightmare to debug, especially if you haven't looked at it for a while.

Tables are a great way to lay data out. People are used to seeing information in rows and columns. They are an essential and powerful tool – but don't try to do too much. Keep it simple.

Strong and Bold Tags

Here I introduce semantic HTML – a major advancement over HTML 4.01.

Take the following:

This bold text is in strong tags.

This bold text is in b tags.

This bold text is in strong tags.

This bold text is in b tags.

They both make text bold. In terms of presentation they are identical. So why have two elements that do the same thing?

They may look the same to humans, but the web crawlers or bots – that crucial second audience – see something quite different. When a search engine spiders and analyzes a page, text in tags is considered important. Text in tags is not. *

Say you are making a page on "Digitigradient". I can't imagine why you would want to, but if you did you might include the following:

Digitigradient

verb – To walk on your toes like a dog or cat.

By putting "Digitigradient" is in tags you are telling the crawlers that "Digitigradient" is important to the meaning of the page while "verb", in tags, is not. Placing "Digitigradient" in strong tags gives it "strong" semantic weight.

The dictionary defines semantics as "the study of meaning". On the web semantics puts meaning into HTML – meaning search engines can use to evaluate a page's SEO.

Semantics is not new to HTML5. Web designers have always had a degree of control of what the bots did and didn't judge significant. For example, anything in <h1> tags has been considered more important than something in say <h6> tags, but HTML5 takes semantics to a whole new level.

One of the biggest changes in HTML5 over previous versions is the addition of the new "section" elements: header, nav, article, footer, aside and section. To site visitors they work the same as tags (which we study next, followed by the section elements), but, like , is semantically neutral. The new section elements give meaning to their content.

Unfortunately just what that meaning is not all that clear – at least not yet. I think we can assume something in the header carries more weight than something in the footer, but I have yet to find any rule where that is stated explicitly. **

My guess is that with time, usage will establish what means what. Incorporating semantics into HTML5 is a good idea – an idea that needs to mature and develop, but a very good idea nonetheless.

There is no need to wait to take advantage of HTML5 semantics, even if it is a bit vague. You can use the strong and b elements (in strong tags) as I suggest and adopting the new section elements now will make your sites forward compatible for years to come. With one small caveat (see below ***), the new section elements are compatible with all modern browsers.

Welcome to the murky waters of web semantics. Keep in mind that you are learning HTML5 while it is still in the design phase – and now is the time to do so, but any software "in beta" has kinks in the works.

At any rate, you can make text bold with all the confidence in the world.

DIV Tag

All the elements we have looked at so far have had a specific purpose. The <p> tag is for paragraphs, <h1> through <h6> set priorities, the <a> tag is for links, etc. The tag, as W3C puts it, is a "generic container for flow content that by itself does not represent anything".

It sounds useless – not to mention dull, but it has developed into a real workhorse. The tag has been the principle tool used to put pages together.

Pages have parts – sections that serve particular functions – and as a rule those parts have been set with tags. On the top of most pages is a header. Likewise most pages have a nav div and on the bottom a footer. These are divs that tend stay the same throughout a website. Then there are divs (or a div) with content that is unique to each page.

In HTML5 there is a <header> element, as well as a <nav>, <footer> and a couple other new elements that replace these div tags.

An example of the header, nav and footer elements

Why study an element whose primary function has been superseded by new elements?

First of all in terms of layout the new elements work essentially the same as divs. What you learn here is applicable to the new elements.

Secondly the tag is still a handy tool. It can do a lot more than just page layout.

Third, as you surf the web and view the source codes of other sites you will rarely see these new tags in use – at least for the time being. With the introduction of IE9 they now have broad browser support. They have worked in Chrome and FireFox for years. However at this point few web designers know they exist, much less use them. My guess is that div tags will be used for page layout for years to come. I hope I'm wrong – the new elements really are a good idea – but at least you'll know better.

Getting back to the tag...

All elements have various properties and all properties have default values. Those default values can be reset by the web designer. How that is done is through the use of Cascading Style Sheets or CSS.

While a full understanding of CSS is beyond the scope of a beginning HTML tutorial, to learn basic HTML you need to know what CSS is and have a general idea of how it works.

Just as adjectives modify nouns, "styles" modify elements.

Take the sentence: "It was a dark and stormy night".

The noun night is modified by two adjectives, dark and stormy.

Were this CSS one would say that night has had its property for illumination set to dark and weather set to stormy. Each property and its corresponding value make a "style". Multiple styles can be grouped as a set, named, and saved as a "class".

The following:

...produces:

  This div tag

      contains this div tag.

This div tag

contains this div tag.

In and each div tag's styles are "modified" according to the styles I set on lines 21 through 24 of the source code. Amongst other things in the outer div I reset the style for the background color (the default value of which is white) to a light grey and darker grey in the inner div. Both have a black border (default is none), but of different thicknesses. There are many styles that can be set to all kinds of different options.

Hopefully you aren't completely lost. Keep in mind that while CSS is a powerful tool, it isn't easy. To make matters worse different browsers do not handle all elements or CSS styles the same way. Over the years Microsoft's Internet Explorer has been particularly problematic.

The good news is that it looks like IE9 is better behaved than previous versions. Better yet, IE usage has dropped from 85% in 2002 to less than 25% now**. FireFox and Chrome have taken the lion's share of the market.*** They're better browsers. With Microsoft no longer the big bully on the block cross-browser compatibility issues should become less problematic. One can only hope.

```
<style>
 .outer {
   width: 960px;
   color: navy;
   background-color: pink;
   border: 2px solid darkblue;
   padding: 5px;
 }
</style> *
```

The style tag is used to create classes. In this case the class "outer" is made up of various styles (width, color etc.) each separated by a semicolon.

This class is then applied to a div tag. Change the body to this:

```
<div class="outer">
  <p>My fourth webpage!</p>
</div>
```

Save it as "my-fourth-webpage.html". Again, you will find a copy of the new file in the source code.

Just so you know, rather than using a class statement you could have written:

```
<div style="width: 960px; color: navy; background-color: pink; border: 2px solid blue; padding: 5px;">
  <p>My fourth webpage!</p>
</div>
```

This is called an "inline style statement". While occasionally handy, it's not nearly as versatile as a class. I'll show you why in a moment.

Save that file and open it in your favorite browser. It should look something like this:

Your fourth web page

Let's take a look at the styles.

width: 960px;

The "px" stands for pixels. A pixel is a size of one dot on a monitor. 960 pixels fits comfortably on a 1024 pixel wide screen **, leaving room for the slide bar on the right.

color: navy; & background-color: pink;

The former sets the color of the font inside the div tag. The latter obviously sets the background color.

border: 2px solid blue;

This sets the border to 2 pixels wide, makes it solid (as opposed to say dashed) and colors it blue.

padding: 5px; ***

This adds 5 pixels of padding from the border inward.

You can play with the styles in my-fourth-webpage.html, but do not be surprised if something doesn't work. I have tried to select styles that are fairly straightforward, but even so your changes may not (often won't) work. Just go back to what I wrote and try something else. Have fun, but don't get discouraged. Eventually it will all make sense.

Let's create another class.

```
.c {
  text-align: center;
}
```

Add the class to the h1 tag: <h1 class="c">My fourth webpage!</h1>

Save it, refresh your browser and you should have:

Your fourth web page

Note that I chose to name the class "c". I could have named it "center" or even "callipygian" **** had I been so inclined, but "c" is easier to remember ...and spell.

The advantage a class has over an inline style statement is that it is reusable. You set a class once and use it over and over again. Above we used a style tag, but better still you can save all your classes in what is called a style sheet (a separate file with a ".css" extension) and use those classes anywhere on your site.

In the head of this page's source code you will see where I call my style sheet:

```
<link rel="stylesheet" href="mycss.css">.
```

Take a look at it. I doubt you will understand much, but at this point you don't have to. At least you know what it looks like.

I have created a fifth page. Open it, copy the source code and save it with the rest of your webpages.

On that page the various divs have different background colors so you can see where each goes.

Now remove the color and the background-color for each class, save it and voila! ...you see div tags in action (actually they should be invisible, but you get my point).

We covered a lot of ground in the last two pages – so much that you may feel a bit disheartened (especially considering what a boring page you've ended up with) but I hope it has begun to make some sense. This stuff isn't easy. Don't worry, it will get better ...eventually.

When all is said and done div tags are not that difficult. Understanding how CSS works can be problematic, but the div tag itself is not. In practice it's really nothing more than a container – a box with CSS styles defining size, color, position, border etc.

Header Tag

Just as a home has rooms devoted to particular activities (ie. one cooks in the kitchen and parks in the garage), there are sections of webpages that have distinct functions.

Since the beginning of the internet virtually all sites have had a "header". On this site, as with the vast majority of sites, the header is at the top of every page. There you should see what the page is about – what it does. This practice has become so widespread HTML5 has a new <header> element to replace the div tag that has largely filled the role so far.

Note that this is an entirely new element with a different purpose than either the "heading tags" (<h1> to <h6>) or the <head> tag. Yes, it is confusing.

This is one of a number of HTML5's new section elements, an important part of HTML5 web semantics.* They are "sections" in the same way that the marketing or human resource departments are "sections" of a business. Such tags define what the different parts of a page's content are and how those various parts are related. As time goes on virtually all pages will have <header>, <nav>, <article> and <footer> sections at a minimum.

No doubt you have heard a great deal about how HTML5 is going to revolutionize the internet – of the exciting and powerful new elements that are going to change everything. It's all true, however the header element is too workaday to foment much passion. Sorry, we will get to fun ones eventually ...but not yet.

We are going to start with something that may lack the hoopla, but is much more important: traffic.

It's a sad fact that there are literally tens of millions of web sites (many some poor sucker paid a fortune for) with all kinds of bells and whistles that virtually no one ever goes to. Their problem is that they don't show up high enough in search results. They rank poorly with the search engines due to bad SEO. The simple truth is that a web site with no traffic is worthless.

As you create sites always keep in mind the fact that websites have two audiences. One is human, the other web crawlers or bots, such as Googlebot. Humans are good at assessing web pages quickly – especially when properly designed.** However it's not so easy for the bots. The header element tells the crawlers what's in the header. Until now they have had to guess.

In other words, these new elements help the bots see sites more like people. The bots can now evaluate the relevance of blocks of text based on the element in which they are found. At the risk of oversimplifying something quite complex, anything in the header carries more weight in terms of SEO than it would were it in say, the footer.***

The search engines want to deliver the most relevant search results first. You want that to be your site. The header tag and the section elements you'll study on the next few pages are tools you can use to help Google and other search engines do just that.

NAV Tag

Like <header>, <nav> is a section element with a clear purpose.

I have it on the left on this site though it's often placed horizontally above or below the header. Technically you can place it (or them – there can be more than one) anywhere you want, but remember never make it difficult for visitors. One of the most common mistakes beginners – and for that matter, many professionals – make is to not make navigation simple, straight forward and intuitive.

The <nav> element is for "major navigation blocks"*. It can go in the header or article tags (which we will look at next); or it can be on its own. On the left I have it on its own and in the article element the "previous" and "next" links are in nav tags.

It's time to get a real HTML editor. There are a number of them. Google "free HTML editor" and see what you come up with. For Windows I like CoffeeCup's free version of its well-known HTML editor. I recommend you download and install it.

For Mac I found TextWrangler. I don't have a Mac so I haven't tried it myself, but a number of people have emailed me recommending it.

If you do google "free HTML editor" you will find what are known as WYSIWYG editors – "WYSIWYG" being an acronym for "What You See Is What You Get". I strongly recommend against them. There are many problems with WYSIWYG editors, but their primary fault is that they do not make SEO friendly pages. They write messy code. It may look good on a monitor, but the bots, your second (and equally important) audience, have a hard time discerning what's what.

No WYSIWYG editor is able to identify what belongs in say the header, nav or footer – that is something only a human can do, therefore they are unable to utilize HTML5's new structural elements – such as the <nav> tag. In fact until now CoffeeCup has been a WYSIWYG editor, but with HTML5 becoming the new standard they are abandoning their WYSIWYG design option altogether.

Assuming you are on a PC**, open CoffeeCup and take a look at a screen you may well find yourself spending a ludicrous number of hours staring – and swearing – at. Needless to say, you can use any editor you please; CoffeeCup may not be for you.*** Choose carefully as web designers get oddly attached to their editors. Whatever editor you do choose may be with you for years to come.

The new file tab in CoffeeCupIn the upper left you will see an icon for "New". Click it, select "New HTML Page" and a new window will open that looks something like this:

```
<!DOCTYPE html>

<html lang="en">

  <head>

    <meta charset="utf-8">

    <title></title>

    <meta name="description" content="">

    <meta name="keywords" content="">
```

```
  <!--[if lt IE 9]>
    <script src="//html5shim.googlecode.com/svn/trunk/html5.js"></script>
  <![endif]-->
 </head>
 <body>


 </body>
</html>
```

Compare the above to what CoffeeCup has as a default template and you will see a couple of differences:

<title></title> should go under <meta charset="utf-8"> and above everything else in the head.

It's not critical, but that is its customary placement.

<meta name="generator" content="CoffeeCup HTML Editor (www.coffeecup.com)"> isn't necessary.

This is CoffeeCup's plug. I guess we can't begrudge them that; they are giving us the software after all.

Nor do you need <meta name="created" content="(date)">

This records when you first created the page. It does nothing for SEO, plus it's information I personally have never needed, but it's up to you.

I might add that there are all kinds of things I don't do that others swear by and some things I consider important that others don't. HTML has few absolutes. You can and should develop your own way of doing things.

If you were wondering, the following:

```
<!--[if lt IE 9]>
  <script src="//html5shim.googlecode.com/svn/trunk/html5.js"></script>
<![endif]-->
```

...tells IE browsers older than IE9 ([if lt IE 9] means "if less than IE 9") to load a javascript that makes the new HTML5 structural tags function. Microsoft's IE didn't support them until IE9.

This is an example of what is called browser sniffing. With the introduction of IE9 Microsoft is finally conforming to basic HTML5 standards. With any luck browser sniffing will eventually become a thing of the past. At least it looks like browser compatibility issues might become less a pain in the neck than it has been to date. I do hope so.

The code tab in CoffeeCupLet's try out your new editor. In the code editor place your cursor between the opening and closing body tags.

Click "code" tab on the left and out will pop a list of all the elements.

Select the HTML5 list of elements.

Scroll down and double click <nav></nav>.

Click the return button on the left and list of elements will slide back.

With the cursor inside the nav tags, repeat the process again only now double click and then inside that double click . In that write home.

And you should have a line of code like:

```
<nav><ul><li><a href="http://www.html-5-tutorial.com/">home</a>
</li></ul></nav>
```

...which you can clean up with "Enter" and "Tab" keys to:

```
 <nav>
 <ul>
  <li><a href="http://www.html-5-tutorial.com/">home</a></li>
 </ul>
</nav>
```

Now I would add <header></header> above the nav tags and include some text in <h1></h1>, clean it up and you're off to the races.

I have created a page, My New Template, based on what you just did. There is also some CSS for page layout. Take a look at it and see if the contents of the body tag looks at all like yours. I named my template "new-template.htm". You should name yours "index.htm" (or index.html) and save it in a folder named "website" on your desktop, or somewhere you can find it later.

Remember, there is no "correct" way to layout a page. I made this template this way to be simple and easy to understand, however I'll be the first to admit it's boring. Be creative.

Keep layout and navigation simple, straight forward and intuitive ...but be creative.

In web design there are some things you can't do and others you probably shouldn't. As you gain experience you will make mistakes – and you will put a lot of time and effort into correcting them ...but then there are those crazy ideas that pop in your head that break all kinds of "rules", but that make a site unique and special – and a joy to make.

The article element

The article element tag

While the contents of the <header> and <nav> elements tend to vary little from page to page the <article> element contains content unique to each page. It's where the action is. What you are reading now is in the <article> element ...perhaps in this case there's not a lot of action, but you get my meaning.

I have spent some time looking into what the experts say the <article> element is and there is not a lot of clarity on the subject. The only relatively consistent thread I can glean out of all of the esoteric talk is that it should contain content that stands on its own the way an article in a magazine would.

Open CoffeeCup (or whatever html editor you use) and open the new template you saved in the previous lesson. If needed open my copy of the new template, copy the source code, paste it into a blank page in CoffeeCup and save it as index.htm somewhere you will find it later.

Now copy the following:

article { width: 950px; display: table; }

and paste it above the </style> tag.

Next copy:

```
<article>
  <h1>The article title</h1>
  <p>This is the contents of the article element.</p>
</article>
```

and paste it under </nav>.

I have a copy of what the new template should look like now with the addition of some back ground colors so you can see what's what. Of course you may have made additional changes than just what I suggested. If so, great – at this point in the tutorial you know enough to do so – in fact play with it to your heart's content. If anything you do doesn't work you can always get back to my copy.

In web design the first thing you want to do is get people on your site. Then once there you want them to stay there, in other words you want your site to be "sticky". Doing so almost always has to do with having pertinent and interesting content. Good content is what visitors want. Give it to them.

Writing content for the web is tough because, unlike a book or a Kindle, reading on a monitor is difficult. It's hard to keep track of where you are when scrolling up and down the page. You will notice that on this site I often break up pages with:

lists

bold or

italic fonts and

examples of <code>

These gives visitors visual clues to keep track of where they are. Well placed photos do the same.

The purpose of a website is realized in its content – be it your goal in creating the site or the visitors goal in going there. The bulk of that content goes in the <article> element.

The footer element

The footer element tag

The <footer> element is the last of what I consider the essential section elements.

Most pages should have <header>, <nav>, <article> and <footer> tags. While not required anymore than a house is required to have a bedroom or kitchen, but it wouldn't be much of a house without them. Likewise these four elements serve distinct functions the vast majority of web pages need filled.

The header, nav and footer tend to stay the same on every page. Once they are set you can copy and paste them into new pages then add the article. Keeping a consistent look throughout a site means visitors can focus on what brought them there in the first place.

The footer can be used to do any number of things that include (but are not limited to) the following:

Use it to complete what was left undone in the header and nav.

A well made header and nav section should be simple and straightforward, but simplicity can come at the cost of details – often important details. "Hana's Hotel" might be enough for a header as long as the footer has "Hana's Holiday Hotel. Your home away from home in the heart of Honolulu".

Often larger sites have links in the footer that don't fit in the nav or if they did fit would distract visitors from more important pages.

Unlike books, websites are seldom read from beginning to end.

Visitors often land on pages other than the home page and proceed through a site in no particular order. This means that in addition to text (that on a web page would go in the article tag) each page has to have the equivalent of a book's title page (the header), a table of contents (the nav section), and a footer that does pretty much everything else.

Visitors like to know not just what a site is about but who is behind it. It's a question of trust.

The web, like any media, is a mode of communication. A website is nothing more than you (or whoever hired you) saying something to anyone who wants to listen. The footer should include information on who "you" are – especially when money might change hands. People want to know all they can about who they are giving their money to.

Include enough information in the footer so visitors have at least a general idea of what's going on.

Many visitors to a site know absolutely nothing about the subject at hand. I once saw a beautiful Bed & Breakfast online that didn't have their location in the footer. It could as easily been in the mountains of New Zealand or Scotland as it could the Rocky Mountains of North America. No one is going to waste their time searching for information that should have been clear from the beginning.

What is in the footer effects SEO.

Information that isn't readily available to people, isn't readily available to the bots. Googlebot can't properly index the Bed & Breakfast site mentioned above without a location.

Writing a good title is the most important thing you can do for SEO but a good title alone is not enough. The contents of the page, including content in the footer, should have information related to what's in the title and description meta tag.

People choose to go to a particular site for a reason and if that site doesn't resolve their issue – doesn't, if you will, "scratch their itch" – they can easily go somewhere else. A good footer can do a lot towards keeping visitors from looking for greener pastures.

Before we go on to the next page let's update the template. Open your copy of new-template-article.htm and try to add the footer. The footer I have in mind is just like the header except that it goes on the bottom of the page. Try to add the CSS as well.

You can see what I did in new-template-footer.htm. There is nothing saying you have to do the same. In fact if you asked ten experienced web designers to write new-template-footer.htm from scratch, odds are that you would see ten different source codes. In particular the CSS would be different. There is no "right" way – which I personally think that is one of the great strengths of HTML and CSS.

The aside element

The aside element tag

My definition of the aside element would be "information either unrelated or loosely related to the main content of a page should go in aside tags" – however, that is not W3C's definition. They write: "The aside element represents content that is tangentially related to the content that forms the main textual flow of a document." *

That confuses me. Anyone who speaks English knows what the word "aside" means. Anyone should be able to more or less understand what the aside element does without having read a word on the subject. After trying to penetrate W3C's definition I'm not so sure. It bewilders more than it enlightens – frankly, it's pedantic nonsense.

The OED defines aside as "to one side; out of the way". ** That I get.

W3C goes on to say: "The element can be used for typographical effects like pull quotes or sidebars, for advertising, for groups of nav elements, and for other content that is considered separate from the main content of the page." *

That's a little better. I don't remember ever using "pull quotes", "sidebars" or "groups of nav elements" and I doubt I ever will. You never know, but I doubt it. On the other hand where to put advertising is a real, everyday problem I and tens of thousands of other web designers do have.

For better SEO I want to be able to separate ads from content – to, if you will, "put them aside". The search engines should know what I think is and isn't significant. The aside element gives me the means to include something in a page that is not important to the meaning of the page. Ads are not related – tangentially or otherwise – to anything more than me needing to pay the rent. They go in aside tags.

I genuinely appreciate the monumental effort W3C is putting into rewriting HTML, but I do wish they would write better English. "Anyone who thinks clearly can write clearly, about anything." William Zinsser, On Writing Well ***

If someone from W3C ever reads this please buy Zinsser's book and read it. In the mean time the rest of us will just have to forge ahead as best we can. Unfortunately there is a lot more of the same where that came from.

The section element tag

The <section> element and the elements we have covered in the last five pages are examples of what are known as semantic HTML. These elements add meaning to content the search engines can use to better understand a page. This page has more on web semantics.

The W3C says: "The <section> element represents a generic section of a document or application."

It goes on to say: "The <section> element is not a generic container element."

I too was confused; however there is a difference.

If all you want to do is have something look different use a div tag (a generic container) and CSS styles.

If you want a particular part of a page separated from the rest of the page due to the information in it - its "content", use section elements (a generic section). Use the section element if there is what W3C calls a "semantic" distinction.

I am still not clear why one is a "container" and the other a "section" – but whatever. Ours is not to wonder why... What's important is that you know when to use one or the other.

You can apply CSS to the section element just as you would a div, however the search engines ignore div tags – they are semantically neutral.

The section element shouldn't be used save as a last resort and then usually it should have a header (h1 to h6). It's like that extra room in the house you never use except when you can't think of any place better. Some working with W3C in the development of HTML5 thought the section element wasn't necessary – they thought it better to just use article tags and at first I was inclined to agree. Then I found the perfect use for it.

On the bottom of this page I have a definition of "Semantic HTML". If you look in the source code (line 111) you will see I have it in section tags. I don't want to lose SEO weight with the aside element, but nor do I want it to detract from what is in the main article tags. The section element fits the bill perfectly.

All these new tags: header, nav, article, footer, aside and section are as a group called section elements. This does not mean the the section element is anything more than just one of them, though you can be excused for assuming otherwise. It took me a while to figure that one out.

Throughout this tutorial I have repeatedly mentioned that you should always be designing for two audiences, one human the other bots. The header, nav, article, footer, aside and section elements are for that latter audience.

These elements give meaning to content the search engines could not ascertain without them. At this point (April 2012) few web designers use them, but they should. They have broad browser support and they make good sense. The better the search engines can "see" and analyze a site the better.

The img or image element

The <img> element, like most elements, is a container. It is not an image in and of itself, but a receptacle for one.

Just as the p element holds a paragraph, the img element holds an image. However, it does so in a entirely different way. Most notably, the image is a separate file that loads into the space created by the img element.

There are three types of image files used in HTML, indicated by different filename extensions: .jpg, .gif and .png (also know as JPEGs, GIFs and PINGs); which we will look at next.

Let's first take a look at what I used to display the image below. Note that there is no closing tag. It's one of the few elements that doesn't use them.

<img src="images/atitlan.jpg" class="flr" alt="Lake Atitlan, Guatemala" width="300" height="240">

Lake Atitlan, Guatemala

Lake Atitlan, Guatemala C.A.*

The img element has a number of what are called attributes: **

The "src" or source attribute tells the browser what image (ie. image file) goes in the tag and where to find it. It is required.

The browser sends instructions to the server to go to the folder named images, get atitlan.jpg and send it back to the client (ie. your computer) and show it in the space reserved by the image tag.

The img element can have a "class" attribute just like most elements.

You can position images, add borders, margins and much more with CSS.

The "alt" attribute refers to alternative text or text that the W3C calls "fallback content". This too is required.

If for some reason visitors can't see an image the alt attribute explains what the image is. The principal visitors that can't "see" images are the web crawlers sent out by the search engines. As I have said repeatedly throughout the tutorial, sites have two audiences – the crawlers being the second. The alt attributes plays a role in SEO.

While the "height" and "width" are not required, they are highly recommended.

Pages don't load all at once. When a new page opens a browser will start to load what it can, when it can and often images finish downloading last. With the height and width attributes set the browser can position (with CSS) and size the img tag before the image file loads.

In the source attribute there are two ways to show what is known as the "path" to the image file:

The "absolute path":

<img src="http://www.html-5-tutorial.com/images/atitlan.jpg" ...

The "relative path":

<img src="images/atitlan.jpg" ...

The latter being "relative" in the sense that its location on the server is relative to the location of the html file requesting it, which in this case is: image-element.htm – in other words, this page.

The relative and absolute paths to atitlan.jpg

On my server is a folder (also called a "directory") named "html-5-tutorial.com" – the "root directory". In the root directory is the file image-element.htm and a folder named images, which in turn holds atitlan.jpg.

The file system on a server works just like the file system on your computer save for the fact that it can be accessed from internet. Your computer has a firewall (or at least it should) that blocks anything from the outside getting in.

It is essential that you understand how paths work as it applies to all files used on any site. CSS, javascript, PHP – what have you – are all stored in files and to access a file of any type you have to follow a path. Even the URL of this page: http://www.html-5-tutorial.com/image-element.htm (or any other page on any site) is just a path. We will work more with paths as we get further along in the tutorial. If you are confused all I can say is not to worry. It might well get worse before it gets better, but it will get better ...eventually.

It was the first time I made a page with photos that I really got hooked on web design. All of a sudden my fledgling efforts actually produced something I could show the world.

As you no doubt know there are many kinds of image files. Images edited in PhotoShop are saved as ".psd" by default, Microsoft's Paint saves images as ".bmp" and so on.

Images on the web are one of three types: JPEGs, GIFs and PINGs – each of which we will look at in turn.

JPEGs (.jpg), GIFs (.gif) and PINGs (.png) files are much smaller than others as they are compressed (or "zipped") by sophisticated algorithms.

How the <img> element works is unaffected by the filename extension (ie. .jpg, .gif or .png). It simply downloads whatever file you tell it to. However the extension does matter to the browser as it tells the browser which algorithm to apply to decompress ("unzip") the file – the reverse algorithm the images editor employed to compress the file in the first place.

Lake Atitlan, Guatemala

Lake Atitlan, Guatemala (atitlan.jpg – 10.2kB)

To compress an image you just save it as a JPEG, GIF or a PING. Your image editor implements the compression algorithm when saving the file. JPEG applies lossy compression, GIF and PING apply lossless compression.

"Atitlan.jpg", the photograph to the right, is 5% the size it would be were it saved in PhotoShop, "atitlan.psd" (211kB) or in MS Paint, "atitlan.bmp" (210kB).

This is important because "atitlan.jpg" takes 5% the time to download than it would saved as .psd. One of the biggest mistakes web designers make is creating pages that take too long to download. People do not want to wait. Bore them for an instant and they're gone.

Lake Atitlan, Guatemala

An overly compressed version of atitlan.jpg. (atitlan-resized.jpg 5.5kB)

JPEG

Lossy compression works best for photographs. At 10.2kB, atitlan.jpg (above) loads quickly while still looking good.

There is always be some loss of detail (hence "lossy"), but don't apply too much compression or you end up with an image like what you see here. You do save 4.7kB in file size, but the tiny gain in page load speed is not worth the loss in quality.

However, apply too little compression and the image file is unnecessarily large. You are wasting server bandwidth, not to mention your visitor's time.

With the correct amount of compression you end up with a photo that for all practical purposes is identical to the image before it was zipped.

GIF & PING

In general, use lossless compression on anything other than photos. Logos are almost always GIFs or PINGs.

HTML5 logo saved as a GIF

An example of a GIF. (html5-badge.gif – 1.2kB)

To the right you can see the HTML5 badge saved both as a GIF and a JPEG. While I over-compressed the JPEG version to make a point, there is always a degree of distortion that comes with JPEGs, while GIFs and PINGs are clean and sharp.

HTML5 logo saved as a JPEG

The same image saved as a JPEG.

(html5-badge.jpg – 1.1kB)

The photo of Lake Aitlan saved as a GIF.

The photo of Lake Atitlan saved

as a GIF. (atitlan.gif – 17kB)

On the other hand, GIFs tend not to be good for photos. When I save the lake photo as a GIF it looks like this. JPEGs can utilize millions of colors while GIFs are limited to a maximum of 256.

As for GIFs versus PINGs, I use GIFs more – but I'll admit to being old school. GIFs were around well before PINGs and I got used to working with them way back when.

In theory PINGs are better than GIFs. In practise browsers (in particular IE – no surprise there) and image editors have been slow to fully support PINGs. Many competent web designers might disagree, but I recommend learning GIFs first.

Photo with transperant gif

A transparent GIF on top of a JPEG.

There is one other important advantage to GIFs and PINGs in that areas in the image can be set to be transparent.

To the right you see atitlan-front.gif on top of atitlan.jpg. When I saved atitlan-front.gif in my image editor I made all of it transparent except for the text in dark blue. Then with CSS I set atitlan.jpg as the background image. You can see the CSS I used to do this in the source code of this page.

In image editors the use of the word "resize" has confused many – myself included. When you want to "resize" an image you would logically think you were changing an image's dimensions, its height and width; not so – at least not in PhotoShop, CorelDraw and a number (but not all) of other image editors. Changing height and width is what they call "resampling". * They call "resizing" compressing the file with one of the compression algorithms (ie. saving it as a JPEG, GIF or a PING).

Lake Atitlan, Guatemala

Here I "resampled" atitlan.jpg down from 300 pixels

wide by 240 pixels high to 150px by 120px. (6.2kB)

"Resample" seems an odd word to use, but it's better than using "resize" in the same context with two different meanings. I don't have a better word, so "resample" it is ...except in those image editors that do use "resize" in the same context with two different meanings.

This is not the place to go into great details about how to edit images. I had a friend once tell me all you had to do to get a photo web ready was resample it and bump up the contrast. He's a nice guy, but not much of a photographer. I also have friends who spend hours agonizing over every pixel. They are great photographers, but don't get much work done. Find an image editor you like and put the time into learning how to use it properly – but efficiently.

There are those that swear PhotoShop is the only way to go – and with some reason. It truly is a powerful program, but it's expensive and difficult to learn. If you know PhotoShop you are way ahead of the game – but if you don't there are all kinds of options, not the least of which are some quite good editors you can download and install for free.

Windows:

PhotoScape – This editor has a weird interface (for example the save button is on the lower right) but once you figure it out it's quite powerful.

IrfanView – As an image editor IrfanView is limited. However, it may well be the handiest little program ever written. I can't imagine owning a computer without it.

GIMP for Windows – I got an email from someone recommending it and I liked it. It's quite powerful.

Mac:

GIMP for Mac – I am not a Mac guy (if for no other reason than that there is no Mac version of IrfanView) but I read some reviews online and GIMP seemed to be the most popular free Mac editor. If you have a better suggestion please contact me.

Images are an essential part of any website. A good working knowledge of a competent image editor is important, but there is a lot to learn – just as there is with HTML and CSS. Get the basics down and with time your skills will improve. For now, let's get back to some HTML.

The map and area elements - image maps

If you click the logo at the top of this page you go to my home page. To do that I simply put an image tag inside anchor tags.

<img src=".." alt="..">

The whole image is a link, but there is also a way to assign different links to distinct areas of an image through the use of a client side image map. *

It sounds geeky – alright, it is a little geeky – but nothing you can't handle. Besides, if you have gotten this far into the tutorial there is a bit of the geek in you. Admit it. It's not so bad. I've lived with it for years.

Below is an image 100px wide by 50px high. Below that is the image tag that displays it.

On Off

<img src="images/on-off.gif" alt="On Off" width="100" height="50">

Here is the process I go through to make that image an image map:

First I add usemap="#on-off" to the image tag.

<img src="images/on-off.gif" alt="On Off" width="100" height="50"

usemap="#on-off">

Second I open the map element with the name attribute.

<map name="on-off">

Third I setup the two areas I want to be "hot":

<area shape="rect" coords="0,0,50,50" href=".." alt="..">

<area shape="rect" coords="50,0,100,50" href=".." alt="..">

Lastly I close the map element.

</map>

There are a few things you need to remember:

The map name must match what is in usemap.

Usemap has a hashmark (#) – usemap="#on-off", but name doesn't – name="on-off".

There can be multiple image maps on a page, but each must have a unique name.

The area element doesn't have a closing tag.

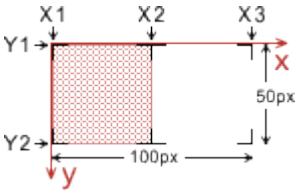
If the area tag has an href="" attribute it should also have an alt="" attribute.

All that looks relatively straightforward except the coords part, which I will admit, is not all that straightforward – but don't panic. Take a moment to close your eyes, take a deep breath and embrace your inner geek.

The x,y co-ordinates of the upper left hand corner of the image is 0,0. Starting there, x counts from left to right and y from the top down.

When the shape is set to rect, coords creates a rectangle. In coords="x1,y1, x2,y2" the first two digits are the x,y co-ordinates of the upper left corner of the box and the second two are the co-ordinates of the lower right corner.

First example of the area element Second example of the area element



U L L R

coords="x1,y1, x2,y2"

coords=" 0, 0, 50,50"

U L L R

coords="x2,y1, x3,y2"

coords="50, 0,100,50"

Below you can see the code in action. Rather that take you to another page I set it so that clicking "1" or "0" below will simply reopen this page, but I could have linked to any page on this site or, for that matter, anywhere on the net.

On Off

Here's the area tags:

<area shape="rect" coords="0,0,50,50" href="map-and-area-elements.htm" alt="Left" title="Left">

<area shape="rect" coords="50,0,100,50" href="map-and-area-elements.htm" alt="Right" title="Right">

Many sites use image maps for navigation.

This is just one simple example of something that can get quite complex. The areas you select as "hot" can be anywhere on the image. You don't have to start in the upper left. The order of the area tags is unimportant.

The shape can also be a circle:

Yes No

<area shape="circle" coords="x,y,radius" title="..">

<area shape="circle" coords="25,25,25" title="Yes">

<area shape="circle" coords="75,25,25" title="No">

or a polygon:

World map

<area shape="poly" coords="x1,y1,x2,y2,..,xn,yn" title="..">

<area shape="poly" coords="2,5,32,1,33,22,51,36,33,57" title="The Americas">

<area shape="poly" coords="57,14,70,2,111,3,114,23,97,34" title="Eurasia">

<area shape="poly" coords="57,14,86,29,73,52,66,49,50,28" title="Africa">

<area shape="poly" coords="105,40,108,49,122,52,127,41,117,34" title="Australia">

There are any number of tools that can help you layout image maps. The image editor GIMP (recommended on the previous page) has a handy tool that does the hard work for you. Polygons in particular can be a real chore without such a tool.

Image maps are not limited to just links. Clicking on an image map button can also trigger a program or script to run. Below I use some basic javascript to open alert windows – however the possibilities are truly unlimited.

On Off

Learning javascript (not to mention PHP, jQuery, Python, ...name your program) is down the road. For now let's not push our luck – you only just discovered your inner geek.

The form element

In general when building a site:

You first want to get people to the site.

Traffic can come from word of mouth, social media, advertising, search engines, etc.

Then you want to keep them on the site.

You don't want them to bounce – to visit one page, not find what they are looking for and go somewhere else.

Finally you want them to do something.

You want them to buy a product, make a reservation, register for something, etc. This is called a conversion.

Getting conversions often entails the use of forms.

No doubt at one time or another you have filled out an online form something like the following:

Name:

Your email:

*

Comments:

+

=

This is so I know a human is filling out the form. *

This form sends you an email. Go ahead and try it.

Below is the opening tag of the form above.

<form action="form-element.php" method="post">

The action attribute tells the browser what to do when the "Submit" button is clicked.

First it sends a message back to the server with the data that was entered into the form.

The server then finds the file indicated in the action attribute – this page/file: "form-element.php".**

It takes the data and does what the instructions in the file tell it to do.

Finally it sends the page back to the client and the browser reloads it.

You will notice that this page has a ".php" extension, rather than ".htm" like most of the rest of the pages on this site. This page contains both HTML and PHP and therefore has a ".php" extension.*** I include PHP code because HTML only displays content. The PHP makes forms function. The instructions mentioned in the list above are written in PHP.

In computer science terms HTML is a markup language, not a programming language**** such as Javascript or in this case, PHP. The browser knows to display the HTML just as it would with an ".htm" extension.

HTML can't multiply two numbers. For that you need a programming language such as PHP. However, you can't display content with PHP. That is HTML's function – with the help of CSS.

x = 0

Let's go through the three steps above to see how this form works.

When you click "Multiply" the browser takes the contents of the two boxes and sends it to the server.

The server then finds the file in the action attribute – again, this page/file: form-element.php.

Then, following a set of instructions I wrote in PHP, the server does the math and saves the answer.

Lastly it sends the data back to the client and the page reloads with the answer displayed in the box.

To generate the answer box above the server sent:

0

But what I actually wrote is:

<?php print $answer; ?>

Now is not the time to learn PHP, but it is important to have a general idea of how HTML, CSS and programming languages work together. There are all kinds of programming languages, and I highly recommend learning at least one, but you should know HTML to use those languages to their fullest potential.

The form element has a second attribute: method="..". The method tells the browser how to send the content of the form back to the server. The post method, used in the previous two forms, hides the data. The get method appends the data to the end of the URL.

<form action="form-element.php" method="get">

To see a get in action fill in your first name and click "Submit".

Your name:

Hello ,

Notice that the URL of this page is now:

http://www.html-5-tutorial.com/form-element.php

You can send that URL in an email or use it in a link on a web page.

http://www.html-5-tutorial.com/form-element.php.

While it's good to know what a get is, unless there are overriding reasons not to, I recommend using the post method. The get method is like that weird screwdriver you hardly ever need, but when you do, it's the only tool for the job.

Unlike TV or print, on the net communication is not a one way street. Visitors can send information back to you via your server. Forms are the principal mechanism web designers use to get clients to do just that. Unfortunately, this opens all kinds of security issues for both visitors and designers.

While many of you filled out the second and third forms, I imagine most of you were hesitant to fill out the first. For all you know I could be some spammer soon to be sending you emails concerning the estate of my late uncle, the ex Nigerian Minister of Finance. I am not a spammer and my uncle sold tractors for a living – but the nature of the net is such that you never know.

Trust is everything. No one with any sense is going to fill out a form without having a reasonable degree of confidence that what they send won't be misused, nor should they. The ongoing – indeed unrelenting efforts of the dead minister's extended family speaks to the fact that there are plenty of idiots out there, but vast majority of us know better.

Web designers have to be careful too. Forms can be vulnerable. I do everything I can to prevent the use of these form for anything other than their intended purpose. Unfortunately, hackers are every bit as unrelenting as the minister's family – and much smarter.

That said, there are ways to protect both your clients and your forms from hackers. Some are low tech. In the first form, as long as I don't save any email addresses (and I don't), no one can hack into the server and steal them. Some are high tech – and beyond my skills to write, however there are free programs you can download and use to process your forms.

The use of one of the three forms above would not be considered a conversion. In fact, this site is unusual in that I am not trying to get visitors to fill out a form; however the ultimate goal of most sites is to build email lists, to request donations, to sell something – to get conversions.

All too often web designers spend too much time on bells and whistles and forget that the point of the exercise is to get people to do something. As a rule it's a simple form on a simple page that brings home the bacon.

The input and related elements

The <input> element is the principal element that collects information entered into a form.

It's been around since the beginning of the web, but in HTML5 <input> has been revamped into a new and greatly improved tool – or at least it will be.

Unfortunately, while W3C has written the new specifications, as of the summer of 2012 the browsers have implemented few of them. Chrome has advanced the most. Go to html5test.com to see how far along in development your browser is.

HTML5 should be finished (what they call a "final recommendation") by 2014 and with any luck all the new input types will be working by then or soon thereafter.

In the form below, except for "email" and "number", all types have been part of HTML for years. Both "email" and "number" work in the latest Chrome.

To see the new types in action open this page in Chrome and fill out the form. Enter whatever you want – you are the only one who will see it. No email is sent to anyone, nor is anything saved. When you click "Submit" the data is displayed below.

Name: required

<input type="text" name="your_name" size="35" value="">

Input type="text" has been the main input "type" used since the beginning and will continue to be so for the foreseeable future.

It has a name attribute, in this case: name="your_name". I will explain the role of the name attribute in more detail below. The tag also has a size attribute and a value attribute, which should be left blank (value="") to display an empty input tag.

The input element does not have a closing tag.

Email: required

<input type="email" name="email" size="35" value="">

Input type="email" is new to HTML5. It's like a "text", but it checks that the content is a valid email address. In Chrome enter something other than an email address, click "Submit" and see what happens.

Age:

<input type="number" name="age" value="">

Input type="number" is also like a "text", but it checks that the content is a valid number.

Gender:

Male

Female

Other

<input type="radio" name="gender" value="Male">

<input type="radio" name="gender" value="Female">

<input type="radio" name="gender" value="Other">

Input type="radio" lets you select one of various option. I include "Other" to show that "radio buttons", as they are often called, can have more than two options. What "Other" constitutes I leave to your imagination.

Notice that all three have the same name and that the "value" attribute is preset.

Games:

Chess

Checkers

Backgammon

Poker

Bridge

Cribbage

<input type="checkbox" name="chess" value="Chess">

<input type="checkbox" name="checkers" value="Checkers">

etc.

With input type="checkbox" you can check multiple boxes. Unlike radio buttons you are not choosing just one of different options.

Again, "value" is preset, however unlike radio buttons, with checkboxes the "name" does change. Take a moment to review the difference between radio buttons and check boxes. As a beginner I found this confusing – but then I am easily confused.

Country:

<select name="country">

<option value="None selected">Please select below</option>

<option value="Afghanistan">Afghanistan</option>

<option value="Albania">Albania</option>

etc.

</select>

While <select name=""> and <option value=""> are different elements, they are related to the input element in that they collect data entered in a form.

Comments:

<textarea name="comments" rows="3" maxlength="200" cols="60"></textarea>

Larger blocks of text are collected by the "textarea" element.

Hidden input

<input type="hidden" name="your_ip" value="">

Input type="hidden" is useful for situations when you might want to gather data not supplied by the person filling out the form. Here I use a hidden input tag to collect your IP address.

<input type="submit" name="submit" value=" Submit ">

Input type="submit" displays as a button with the contents of the "value" attribute written on the button.

Clicking "Submit" triggers the browser to send the information entered in the form back to the server and process the data according to the instructions in file indicated by the action attribute of the opening form tag: <form action="input-element.php" method="post">. This was covered on the form element page.

All elements used to collect data in a form have a "name" attribute (name=""). That is the name of the variable that is assigned the value of the data collected by each input, select or textarea element.

You haven't yet filled out the form and submitted it; therefore the program hasn't run and no changes have been made. Try it and see what happens.

The variables and their corresponding values are sent to my server where they are picked up and processed by the program in input-element.php. The HTML is then reassembled and sent back to your browser with the new changes in place. While this particular form does not, normally there would be instructions in the program to send the website owner an email that included the data entered in the form.

We have covered a lot of ground on this and the previous page. My goal was to give you an overall understanding of the role forms play and general idea of how they work, rather than explicit instructions on how to build a functional form. That will have to wait for a future tutorial. There are a few things you need first – not the least of which is a server.

If you have a server, great; if not, you are getting to the point where you may want to think about getting your own domain and finding a hosting company. Next we will look into doing just that.

What is a domain name?

On March 15, 1985 the first domain: symbolics.com was registered. A year later all of 10 more domains had been registered. Now there are over 100 million of them. The idea caught on.

They say all the really good domains are taken – and I had the chance to get some great ones. I tell myself there are still plenty of perfectly fine domains to be had, that the domain does not play a big role in SEO, that too much emphasis is placed on getting the perfect domain; and it's all true, but if only...

Let take a look at a couple of examples.

shoes.com

Great ecommerce site. It feels clean and secure. The owner of that domain just sits back and watches the commissions roll in.

realestate.com

This guy has both money and taste – probably drives a Lamborghini. But what a spectacular site! Nothing gets in the way of you searching for a home.

While having great domains definitely doesn't hurt, they are not the only – or even principal – reason those sites are successful. I have repeated throughout this tutorial is that the best website in the world is worthless without traffic. The same holds true for domains.

Take a look at sunglass.com. It too is a well designed ecommerce site. However it doesn't show up in the search engines; bad SEO. Google "sunglasses" and what comes up on top is sunglasshut.com – not as good a domain, but a much better business.

Getting a good domain is important, but it's just one part of the process. Find a domain that is easy to remember and descriptive. Do the best you can, but there is not much gain in fretting if it isn't ideal. Focus on SEO and good clean design – things you can control.

So what is a domain? What we think of as a domain is actually a combination of a middle (or second) level domain, eg. "HTML-5-tutorial" and a top level domain (TLD), "com".

The most common top level domain names are com, org and net.

.com is for commercial sites.

.net is for internet related sites.

.org is for non-profit organizations.

However, there are many other top level domains. Countries have country code top level domains. For example India has .in and Brazil, .br. Certain industries have their own top level domain like .travel for the travel industry. *

Like a telephone, every computer connected to the internet has a unique number. Both the computer in front of you and my server have Internet Protocol or IP numbers, called an IP address.

Your computer's IP address is: 193.1.11.130

My server's IP address is: 66.147.244.113

As IP addresses are difficult to remember, a system was created to give IPs names, called domains. Why the powers that be decided to call them "domains" rather than just "names", I have no idea. Domains are something kings rule ...I feel a rant coming on, but I will resist.

The system that manages domains is the Domain Name System or more commonly, DNS. **

The Domain Name System is a database with domains in one column and their associated IP addresses in the other. It is nothing more than an enormous electronic "telephone" book stored in DNS servers around the globe.

To display a website your browser first needs to look up the domain's IP address; in computer science terms, it needs to resolve the domain. Why "resolve" instead of "find" or "look up"? Again, I have no idea...

To resolve the domain the browser requests a DNS lookup from a DNS server. The DNS server replies with the IP address and the browser then downloads the HTML, CSS, image etc. files from the server with that IP address.

Keeping track of tens of millions of domain names and delivering their IP address to hundreds of millions of computers around the globe in a tiny fraction of a second is no mean feat. The DNS's speed and accuracy is the result of some extraordinary technology, but what it actually does is quite simple – indeed, elegantly so.

So how do you go about getting a domain? Domains, like IPs, must be unique, so the first thing you need to do is find a domain that isn't already taken. When I started in web design in the late 1990s, to see if a domain was available I went to Internic "Whois" to see who is the owner of a domain. If whatever domain was not taken it responded with: No match for domain "example.com". I still go there to see who owns a particular domain because I know no one is going to pull a fast one on me. ***

When I am looking to reserve a new domain I go to bluehost, where this site is hosted. I know I am going to get a sales pitch, but I can trust them. ****

If you want see if a domain is available go to: bluehost, click "Sign Up Now" ***** and enter the domain you would like in the box under "I Need a Domain Name". You can continue checking new possibilities until you find a name you like. Once you do, leave that page open, come back here and learn about servers.

Registering a domain requires an associated IP which, in turn, requires having a server – and that you do need to sign up, and pay for. There is an annual fee to register a domain, but with bluehost, and almost all hosting companies I know of, your first domain is included.******

So now the question becomes, how do you get a server?

What is a server and how do I get one?

A rack of servers

Servers at a hosting company.

For all its complexity, the internet is nothing more than a mechanism computers use to share files. A server is a computer that stores and distributes files to other computers – such as HTML and CSS files.

inside a server

Inside a server

Most servers run a UNIX based operating system rather than Windows or OS X. The physical components need to be more robust due to the higher demands placed on them, but it's basically the same machine as what you have in front of you. The only difference of any importance to most of us is that servers are owned by hosting services and rented to people like you and me.

So how do you get a server?

There are all kinds of hosting companies renting space on different kinds of servers.* It can be difficult for a novice to know where to start. The waters are thick with an all but impenetrable slurry of sales pitches and computer jargon. Anyone who knows what they are doing can navigate just fine, but anyone who knows what they are doing is no novice. Add to that the fact that there are hustlers on the fringes not above out-and-out fraud. None of the large hosting services cheat, nor do the vast majority of smaller companies, but you do have to be careful.

As a novice, odds are that all you need is the inexpensive basic package most hosting companies offer. Choosing one company's basic package over another's has little to do with the plethora of features each company offers. For all the hoopla, they all sell essentially the same service.

In my experience the only thing that separates a good hosting company from a bad one is their tech support. Period. The best hosting companies base their tech support people's job performance on client satisfaction – not the number of support tickets they can crank out or even their computer skills.**

I started in web design in 1998 and I have worked with I don't know how many different hosts over the years. As you may have discovered reading through this tutorial, I am not very tolerant of geek talk. This comes mainly from the many many frustrating hours I have spent on the phone with, or trying to decipher emails from, tech support.*** I've heard it all! To this day I will not work with clients who host with GoDaddy or Network Solutions. My blood pressure goes up just thinking about it.

A lot has changed since 1998. Servers are more stable now and tech support has unquestionably improved. I've been told that even GoDaddy's tech support is better than it was. Hosting services in general have improved with time, as well they should have. You might never need tech support – with any luck you won't – but these are complex systems and things can go wrong. If you do need help, you have to know it's there.

Sign up with BluehostThis site is hosted with Bluehost's basic package, and yes (in the interest of full disclosure), I do get paid a commission on sales.**** I have worked with them as a web site designer on and off for years, (long before I started this site), mostly though clients who hosted with them.

I can't say that Bluehost is the best hosting service there is bar none, nor that their tech support outshines the rest. I haven't made that extensive a study. However I can say that Bluehost has been around a long time and that I am a knowledgeable, demanding – and satisfied – customer.

Over the years I have contacted Bluehost's tech support a number of times. They always replied promptly in jargon-free English. Recently there was one minor issue we didn't completely agree on. I wanted something they don't offer, however their explanation for why not was clear and reasonable – fair enough.

If you found a domain you like on the Bluehost site go back to that page and follow the instructions. If not, go to Bluehost, click "Sign Up Now" and search for a domain. Once you decide on one it's time to take the plunge. If you plan to host with a someone other than Bluehost the process is pretty much the same. You should be able to follow what I am doing on the next couple of pages without too much difficulty.

Next I will walk you through each step of the process of signing up with bluehost. Then we'll take a look at cPanel, create an FTP account and last, but not least, make a simple site.

Signing up with a hosting company

If you are signing up with a host other than Bluehost the step-by-step instructions that follow won't apply, however nor should it be all that different. The information they all require is essentially the same as what Bluehost collects:

A domain

Your name, email, etc.

Some money

A login and password

Sign up with BluehostWhile the signup process may differ, the "control panel", which you will need to manage the site, should be virtually identical. A private software company, cPanel, sells Bluehost and the vast majority of hosting companies their software.

In fact cPanel is so popular the "control panel" is almost always called the "cpanel". It's like "hook-and-loop fastener" being called "velcro". If your potential hosting company does not have a cPanel "cpanel", I would consider looking for a new host.

All HTML5 Tags

Below is a list of all html tags with links to their page on this site, W3C.org and www.w3schools.com.
W3C is the principle organization that sets standards for HTML. While I like what they are doing with HTML5, W3C's site is next to impossible to navigate and their language tangled at best. However, they are the powers that be. As a web designer you will eventually find yourself on their site. You may as well start now.
I often use **W3schools** as a reference site, plus they have good intermediate level tutorials. In fact I recommend them as a good site to continue your studies after you have gotten the basics down here.

Below I have links to the appropriate page on this site, on W3C and w3schools. There are a number of elements that do not yet have a page on this site, and therefore don't have links. I will add them as the site grows.

Below you see elements in grey, red and pink.

Elements in grey are in previous versions of HTML, but are not supported in HTML5. [1]
Elements in red are new to HTML5 and have broad browser support.
Elements in pink are new to HTML5, but don't yet have broad browser support.

The tags:

HTML-5-tutorial	W3C	W3schools
<!DOCTYPE>	!DOCTYPE	!DOCTYPE
<a>	a	a
<abbr>	abbr	abbr
<acronym> [2]		
<address>	address	address
<applet> [3]		
<area>	area	area
<article>	article	article
<aside>	aside	aside
<audio>	audio	audio
<b>	b	b
<base>	base	base
<basefont> [4]		
<bdi>	bdi	bdi
<bdo>	bdo	bdo
<big> [4]		
<blockquote>	blockquote	blockquote
<body>	body	body
 	br	br
<button>	button	button
<canvas>	canvas	canvas
<caption>	caption	caption
[4]		
<cite>	cite	cite
<code>	code	code
<col>	col	col
<colgroup>	colgroup	colgroup
<command>	command	command
<datalist>	datalist	datalist
<dd>	dd	dd
<del>	del	del
<details>	details	details
<dfn>	dfn	dfn
<dir>		dir
	div	div
<dl>	dl	dl
<dt>		
<embed>		
<fieldset>		
<figcaption>		
<figure>		
<font>		

HTML-5-tutorial	W3C	W3schools
<kbd>	kbd	kbd
<keygen>	keygen	keygen
<label>	label	label
<legend>	legend	legend
	li	li
<link>	link	link
<map>	map	map
<mark>	mark	mark
<menu>	menu	menu
<meta>	meta	meta
<meter>	meter	meter
<nav>	nav	nav
<noframes>		
<noscript>	noscript	noscript
<object>	object	object
	ol	ol
<optgroup>	optgroup	optgroup
<option>	option	option
<output>	output	output
<p>	p	p
<param>	param	param
<pre>	pre	pre
<progress>	progress	progress
<q>	q	q
<rp>	rp	rp
<rt>	rt	rt
<ruby>	ruby	ruby
<s>	s	s
<samp>	samp	samp
<script>	script	script
<section>	section	section
<select>	select	select
<small>	small	small
<source>	source	source
	span	span
<strike> [4]		
<style>		
<sub>		
<summary>		
<sup>		
<table>		
<tbody>		
<td>		

<footer>	dt	dt	<textarea>	strong	strong
<form>	em	em	<tfoot>	style	style
<frame>	embed	embed	<th>	sub	sub
<frameset>	fieldset	fieldset	<thead>	summary	summary
<h1> - <h6>	figcaption	figcaption	<time>	sup	sup
<head>	figure	figure	<title>	table	table
<header>			<tr>	tbody	tbody
<hgroup>	footer	footer	<track>	td	td
<hr>	form	form	<tt>	textarea	textarea
<html>			<u>	tfoot	tfoot
<i>	h1 - h6	h1 - h6		th	th
<iframe>	head	head	<var>	thead	thead
<img>	header	header	<video> [5]	time	time
<input>	hgroup	hgroup	<wbr>	title	title
<ins>	hr	hr		tr	tr
	html	html		track	track
	i	i		u	u
	iframe	iframe		ul	ul
	img	img		var	var
	input	input		video	video
	ins	ins		wbr	wbr

Helpful links

Below are links to sites I have actually learned something from – many I continue to use regularly. Most assume a degree of skill you might not have yet acquired. This page isn't going anywhere. **Bookmark it** and come back later.

The links below are for free online services.

101besthtml5sites.com

About.com

Internet for Beginners

Color Schemer

Color Schemer Online

css3maker.com

Favicon Generator

feedthebot.com

Googlebot page speed guidelines and issues

Font Squirrel

Free fonts

Google

Google 101

Webmaster guidelines

Google's Let's make the web faster

Google's Search Engine Optimization Starter Guide (pdf)

Google Trifecta for your Website - YouTube Video

Google Webmaster's Tools

Google Analytics

Check if Analytics is properly installed

Google Web Site Optimizer

htaccessredirect.net

Messing with .htaccess can be tricky. This is a handy resource, but be careful. (ie. backup!)

Hotscripts.com

A handy PHP, CGI, Perl, JavaScript and ASP script resource web portal.

HTML5Anchor

The link to all HTML5 resources

Html5Bookmarks.com