**BECOME A WEB DESIGNER**

Today’s web designer balances clear concise design skills alongside a strong foundational background in technology of the web. These tutorial cover skills you need to design rich engaging websites and applications that look beautiful on both desktop and mobile devices.

**Learn:** the fundamental of HTML and CSS web based design

**Practice:** using industry standard modern tools to design websites and applications

**Create:** designs that adapt to fit wide variety of devices and screens sizes.

**CHAPTER 1: INTROCUTION TO WEB DESIGN AND DEVELOPMENT**

A: Getting Started with Web Design:

Web design is a huge field with multiple areas of specialization, and often the term is used to identify anything from creating the visual design of sites all the way through the development of sophisticated web applications. Because of this, it's really difficult to summarize which skill sets someone will need to master in order to become a web designer. Much of that is gonna depend upon the area of web design that you wanna focus on. However, there are some basic core skills that almost all web designers will need to learn.

Regardless of which area of web design you specialize in, you'll need to have a working knowledge of HTML, CSS, and JavaScript. These languages form the core of almost all web content and should be considered as essential skills. Although there are a lot of web design tools that make the process of authoring content easier, none of them allow you to create professional websites without writing code. Regardless of what their marketing teams tell you. And that's not likely to change in the near future either, so if you're serious about learning web design, you'll need to embrace the prospect of learning to code.

Some people tend to have anxiety over learning to code while others just naturally dive right into it. If you're one of the ones that are a little hesitant about it, let me reassure you that coding is not hard, and picking up the basics is incredibly easy. To show you how easy, this first chapter is gonna be a quick introduction into those three languages. We're not gonna do any deep dives, or focus too much on syntax. Rather, I want you to experience firsthand how simple most of the code that's required for websites really is. I found that once people get in, write a few lines of code, and then see it work, all that anxiety tends to melt away.

We'll start our next movie by exploring the foundation of all web pages: HTML. I think it's important to set expectations at the outset here. This is not a hands-on course. Although we'll be doing some basic exercises over the next few movies, this course is more focused on the concepts and approaches of learning web design. At the end of the course, I'll provide links to courses that will allow you to practice the concepts that we'll cover here in an applied way.

B: Exploring HTML

- At the heart of every web page is HTML. It's the standard markup language for the web. A markup language simply means that you're going to be using tags to identify what type of content the browser should display. Let's go ahead and create a basic HTML page so that you have an idea of how all of that works. I'm on a Mac so I'm going to use a program called TextEdit. I'm going to go ahead and open that up.Basically, TextEdit is nothing more than a basic plain text editor. If you're on a PC instead of a Mac, feel free to use Notepad, it'll work just as well.

You can use any plain text editor that you want. There's a reason, actually, that I'm not using a dedicated code editor for the web. If you feel like using one of those, like Brackets, which is the one I'm currently using, or Sublime Text editor or something like that, feel free to go download them. Brackets, for example, is free. However, I'm not using a code editor for this particular series of exercises for two very important reasons. Number one, to convey the concept that HTML files, CSS files and JavaScript files are really nothing more than just plain text files.

That's all they are. They have a certain extension on them which tells the browser how to parse them. Other than that, they're just plain text.It's sort of the beauty and simplicity of the web if you will. The second thing is to show you that you don't need thousands of dollars of tools and equipment to create websites. If you really wanted to do it old school, you could open up Notepad, open up TextEdit and just go to town. They don't have the really robust coding tools that a lot of the other tools have out there, or coding features, but you could get it done.It certainly wouldn't be as efficient, but it would work.

Let me close this, because there's a couple preferences I want to show you. If I open up my preferences, if I go to new document, I want to make sure that regardless of which program you guys are using, whether it be Notepad, TextEdit or something else, make sure that it's creating a plain text document for one. We don't want a rich text document. The second thing is, I'm going to make sure it's not adding any extensions by default. I want to do that. In TextEdit, I have to make sure that it displays HTML files as HTML code instead of formatting the content for me. You might want to make sure that is turned on.

If you're using a program like Notepad, check its preferences and make sure that, if it has similar settings, that you go ahead and set them that way. Now I'm going to go ahead and create a brand new file and then just very quickly I'm going to save this. I'm going to go out to my desktop, find the exercise files that I downloaded and saved, and save it to the 01\_01 directory. Now, if you don't have those exercise files,you can go to the web design fundamentals course page at lynda.com and download those. They're free for everybody. They're very simple files. If you don't have them and you just want to save this to the desktop, that's fine.

I'm going to title this hello.htm and that's all that's required for me to take a plain text file and convert it to an HTML document; that's it. I could also use the extension .html, that would work just fine as well. It used to matter in terms of the server whether you used the htm or html extension. Now it doesn't, it's just more of a personal preference. I'm going to his save, there we go. Now I have an HTML file. There's nothing on it at the moment, but I have one. If I type in something like My First Web Page, If I save this and go out to a browser, and I'm going to just go out to, let's say Google's Chrome browser and I'm going to tell it that I want it to open a file.

I'm going to go to desktop, exercise files, hello, and boom, open that up. You can see, in this tab, in my browser it says My First Web Page.HTML is nothing more than a text file. It's working within the browser right now, but it's not supposed to be. Chrome is actually doing us a favor by rendering this. I've broken all kinds of rules because instead of using a markup language to mark up content, I just placed content on there and then Chrome is going, "Oh, well "it's an HTML file so I don't see what I need to see "so I'm just going to build it for you." It's just going to scaffold all that stuff for me.

As a matter of fact, if I inspect this element, if I right click and choose inspect element, you can see all this HTML code that we didn't writedown here in the WebKit inspector. Where that is coming from is that Chrome's just making it for us. It's like, "You kind of needed this," so it makes it for us. Let's go back and take a look at what's required for the most basic of HTML documents. I'm going to go in and the first thing I'm going to do is at the very top, I'm going to add an html tag. Tags are going to have these angle brackets.

You can see I've got sort of a little less than and a greater than symbol and I'm surrounding the text html. Below the content where it says My First Web Page, I'm going to create a closing tag. This is an opening tag up top and this is a closing tag at the bottom. You'll notice the only difference between them is the forward slash. The forward slash indicates that this document is closing. This is opening a document, HTML file, and this is closing it. The next thing the HTML file needs is a head. Every single HTML file, and I'm just going to go down to the next line here, type in head.

Every single HTML file has a head at the top of it, just below the HTML tag itself. The head of the document is where all of the stuff that goes into making it work that aren't really visual, not visual elements, so links to external JavaScript files or CSS files, meta tags that describe the site, that sort of thing, they go in the head of the document. Here, I'm going to go ahead and put in a title. You're going to notice that I'm nesting tags inside of other tags, so the head tag is totally nested inside of the html tag.

The title tag is going to be totally nested inside of the head tag. This is what we call child and parent tags. Obviously the parent tag is the one towards the top. The child tag is the one nested inside of it. If you nest one tag inside of another one, you have to close it before you can close the parent. When I say title and I do something like Exploring HTML, I have to close that title tag before I can close the head tag.That's just a little bit of syntax, not a lot but just a little bit.

After the head, I have to have a body. The body is where all the visual content of the page goes, and you can see that I'm going to surround that text with the body. If I save this now and go back to my browser and refresh it, nothing really happens except for the fact that in this tab, the Exploring HTML comes up because now that's the title of that. As a matter of fact, if I inspected this element again, and I'm right clicking, by the way, when I do that to choose inspect element, I'm coming in here and I can see within the head now, there's that title.

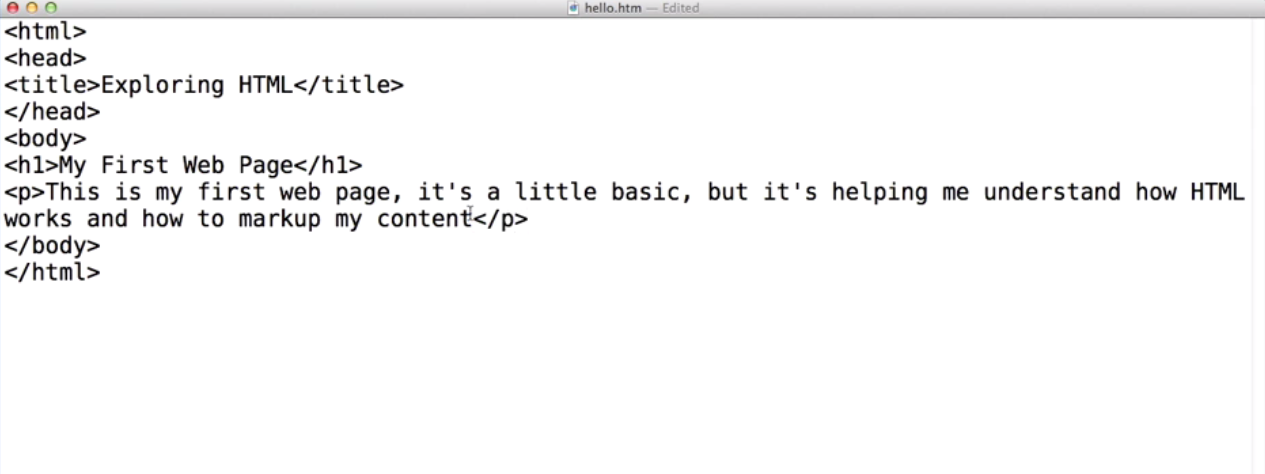
Everything else looks pretty much the same. That's because, remember, before Chrome was sort of doing that for us. Now we have the actual structure there. Of course, markup languages use tags to identify content. Right now, we're not identifying the text My First Web Page is anything other than text that's just sitting within the body. When that happens, a browser is going to use the default tag which, in this case would be paragraph, so it just assumes that that's a paragraph. What if we wanted it to be something else, like a heading, for example? To do that, we would just need to wrap that in a heading tag and we would need to use whichever heading tag we wanted to.

The h1 is the most important heading tag. I'm going to do an h1 tag. Again, I have the opening h1 tag. You can see, I'm using those angle brackets, less than, greater than, and then the text h1. Right after it, I have to close that h1. If I save this and then go back into my browser and refresh it, now I see a physical difference in the rendering of that content because Chrome says, "Oh, that's a heading and "that's a heading 1, that's the most important heading. "I'm going to go ahead and apply my "default styling for an h1 to that.

"I'm going to make it big, I'm going to make it bold, "I'm going to let people know that "this is indeed a heading." It's reacting to the fact that we've identified this content as being a different type of element. I'm going to finish this up by going in and just doing a very quick little paragraph. The paragraph tag, of course, uses a p. You'll notice heading 1 uses h1. The tags are pretty self-explanatory. I bet that most of you, if I just showed you the p tag and said, "What is this?" You might answer, "Oh, paragraph." That might just roll right off the tongue. Most of them do kind of make sense like that, b for bold, i for italics, p for paragraph, that sort of thing.

I'm just going to type in This is my first web page, it's a little basic, but it's helping me understand how HTML works and how to markup my content. Once again, if I open a paragraph tag, I have to close it so at the end of that, I'm going to go ahead and close that.

You know what? It might be nice if I put a period in there just to give that some punctuation. All right, I'm going to go ahead and save that, go back to my browser and when I refresh, there is my brand new content. Now, there is, of course, a lot more to HTML than what we've done here. Really, in only about five minutes, you've just learned the fundamentals of how HTML works.



You've structured a document correctly by adding an HTML, a head section and a body section and you've added and structured the page content correctly as well by using an h1 and a paragraph tag.

Once you have those basics down, it's really just a matter of learning the remaining syntax rules and which elements are available for you to use. Those are the basics of HTML. Let's go ahead and take a look at how CSS is used to control the visual style of page elements in our next movie.

C: Exploring CSS

Cascading Style Sheets, or CSS, is used to control the visual presentation of web pages. I wanna explore some of the basics around its syntax and give you an example of how it works. I have the hello.htm file opened up from the 01\_02 folder, although to be honest with you, it's exactly what we had in the last exercise, so if you still have that open and you just wanna work with that, there's nothing wrong with that either. Just a brief reminder of what's going on within the HTML of our file. We have an opening and a closing HTML tag that signifies the start and the end of an HTML document.

We have the head of a document, and remember, this where all the stuff goes that makes the page work, but it's not visual, you don't actually see it. In this case we have a title, and that's really all we have here, and then we have the body of the page. This is what you'll actually see in the browser. Here, of course, we only have a heading, one, an "h1," and we have a paragraph. So that's really all we have going on. If I look at that inner browser, I can see that we have the heading up top and then the text underneath that. So CSS controls styling, of course, and I'm going to embed this CSS directly inside the HTML document.

This is just one of the ways that we can use CSS. A more common way to use it is to actually create it in its own CSS file and then link to it from an HTML file. But just in the interest of being efficient and being quick, I wanna show you how to do it within the HTML document itself.It's one of those things that helps make the page work, but that isn't actually visual, so that's gonna go in the head of the document. What I'm gonna do is, right after the title, I'm gonna open a style tag, and then I'm going to close a style tag. You can see that's just HTML, it's just like everything else that we've written so far, but the style tag tells the browser, "Okay, inside of this, I'm gonna write CSS." So it gives you that little space inside the HTML document that allows us to write styles.

If I wanted to link to an external document, I'd be using a link tag to bring in that external resource. In this case, since we're dong it within the document-- and this is known as an embedded style-- we're gonna use the style tag to do that. I'm gonna give myself a little bit of room here,and now, the syntax that we're gonna be using is CSS. We don't need inside the opening and the closing style tag, we don't write HTML anymore, now we write CSS. CSS is not a markup language, it's a presentation language. So, its syntax is a little different.

It's essentially really, really easy. The only thing it is is a selector which targets an element on the page, and then rules that say, "I want it to look like this." Let's say we type in "body." Now, this is a selector, so the text body is a selector, and what's it gonna style? Everything inside this body tag. After the selector, I'm gonna open up a curly bracket, and you can find these little curly braces just to the right of the "P" key on your keyboard. I can do it all in one line, but I've typed it this way for so long.

I'm gonna go down to the next line and I'm gonna type in a closing curly brace. And you can see, that's the opening one, it's facing towards the left, and this one is the closing one, it's facing towards the right. Basically, when you have a selector here, inside these curly braces iswhat we call the declaration block. That means, "Style this element, "and here's what I want you to style." And of course, we don't have any properties in there yet, so inside that blank line in-between that, I'm gonna type in the word "color", type in a colon, and then I'm gonna type in the value "red" and then a semi-colon.

So this is a property, and this is a value. They are separated by a colon, and once you're done with one property-value combination, you use a semi-colon to move to the next one. So I'm gonna save that, and then if I go back into my browser and refresh the page, you can see that all the text is now red because everything inside the body is going to be red. Of course, I can be a lot more specific if I want to. I'm gonna go down to the next line and type in "h1." You can see the target and HTML element, you don't need to enclose it in tags, you just tell it which element you wanna style.

It's really, really simple. I'm gonna do the same thing that I did before, which is give myself a little bit of empty space here and have the opening and closing curly braces. The reason that I like to build those first is because I always, if I don't, I always forget to type in the closing one, so that's something that I like to do. Now, we're gonna get a little bit more adventurous here, we're gonna go beyond just color. I'm gonna type in "font-family" and then a colon, and then I'm gonna type in the word "Arial," I'm asking for the font Arial, comma, "sans-serif" and then a semi-colon.

What's happening here is I'm saying, "Hey, the font family, "the font that you're displaying this text in, "I want you to change it to Arial for me."The reason that I have two declarations here, and they're separated by a comma, I'm saying, "If Arial's not available, give me whatever your default sans-serif font would be." Then I'm gonna get into the next line, I'm gonna type in "font-size:" and then let's make this a 32 pixels. So "32px," you'll notice the "px" is coming right after the 32, there's no space between 'em, and then I'm typing in a semi-colon, and then let's go ahead and change the color of our heading, too.

I'll say "color," in this case I'm gonna say "black." There are other ways to define color, but the key words are probably one of the easiest things to learn if you're just gonna use some basic colors. I'm gonna save that, I'll go back into my browser and refresh that. Notice that the heading now changes its styling independent of the paragraph text. It's black, it's Arial, it's 32 pixels, it's exactly what I asked it for. The paragraph remains unchanged, of course, because I only targeted the "h1" element. Now, much like the HTML language, there's a whole lot more CSS to learn.

These are just the basic concepts of what CSS is doing on a page, but once you understand those basics that we just covered, it's really easy to get started learning the rest of it. The syntax is really simple, as you can see. It doesn't get any more complex than this, and for the most part, it's just a matter of learning how to write which selector you need to target specific elements on the page, and then which properties and values that you need to use in order to give you the stying that you need.

Top of Form

Bottom of Form