Transcript

Lecture 10: Website Development

Hour 1

(00:00:00)

DAVID MALAN: Welcome back to Computer Science E-1. This is milk and cookies night, as you may have noticed in back. And also "Website Development."

So two hours hence, you all will know how to make Websites. And to make this all the more of an awesome notion, let's take an arbitrary Website. Tell me your favorite Website. Anything?

STUDENT: (inaudible response)

DAVID MALAN: Okay, that actually lends itself to discussion. As many of you probably saw something like this, or even better, nothing at all for much of today. As of this morning, FAS's Web servers, rather ironically, given tonight's topic, are not working.

Fortunately, we are outsourcing your final projects, as you already know, to a third-party host. So that actually works out well, anyway. But if you've had trouble submitting to the dropboxes, accessing your FAS email, or visiting the course's Website, you may soon be seeing messages like this. Hopefully they'll be gone within a few hours, or certainly by morning.

If you had trouble submitting anything because of this, just email us or call us when you can. Let us know, and we'll work out the details, since this obviously isn't your fault.

I need your second-favorite Website tonight.

STUDENT: (inaudible response)

DAVID MALAN: Whoo! Good one: weatherchannel.com. So weatherchannel.com has today's forecast. And more dramatic than today's weather will be the stuff behind today's weather. So this is just a Website I've pulled up, using Internet Explorer. I've gone up to the View menu, up to Source. And what is this stuff that I've just pulled up, albeit in rather small text?

STUDENT: It's the HTML code.

DAVID MALAN: Yeah, so this is the source code, so to speak, underlying weatherchannel.com. HTML or XHTML are the languages in which Web pages are written today. And you'll often see, coupled with XHTML or HTML, a language called JavaScript, which you see some snippets of above here. You'll see a language called Cascading Style Sheets, a language of sorts, which we'll also introduce tonight. And sometimes you'll see other types of technologies underlying a Web page's design.

The takeaway, though, for this introduction, is that that front page that you yourself might pull up every day... should not have this many blank lines in it. This perhaps could be our example of bad design. Although, let's see, we can scroll over. Yeah, no, it's not even over there. So, by the end of tonight, you will be able to make things that look like this, all right? So if that's not already enticing enough, let's motivate discussion with some questions.

So, a Website, what is it? Sort of a deceptively simple question. What's a Website? Yeah?

STUDENT: A bunch of files that you download.

DAVID MALAN: All right, good. So it's a bunch of files that you download. That sounds awfully like my email, if I download, say, attachments from an email. So how's a Website different from files that I might download from emails?

STUDENT: It's viewable by anybody.

DAVID MALAN: So it's viewable by anybody. So there's this public aspect to it, assuming you don't have usernames and passwords.

The Web itself, we said in, like, Lectures 1 and 2, is a service, really. It's an application of sorts that runs on top of the infrastructure that we said was the Internet.

Tonight's focus, then, will be entirely on how you can make use of that service, the World Wide Web: Put yourself out there, put your business out there, put your photos out there, put your... anything out there in a form that, for the most part, is completely publicly accessible. And next to email, the Web is probably the most popular use of the Internet today, be it to check today's weather, the course's Website, or any number of Websites out there.

Before we proceed too much further, though, I wanted to draw your attention to such handouts as this tonight. So this is a photocopy of an article from *Money* magazine. Sort of strange to be in a CS class, getting photocopies of *Money* magazine. But what they had in their most recent December issue was a little hand-holding article called "Defend Your Virtual Home." And it's essentially a walk-through of all the kinds of things you can do, or should do with respect to your security and privacy on the Internet; all of the things you shouldn't do, or should not be able to do on the Internet, with regard to your security and privacy.

So we offer this tonight, as well the following recap, since we saw some looks of fear and some comments of fear, based on the last couple of lectures, in which a few of you seem to have left sort of more scared about what your computer can do, or what it might be doing unbeknownst to you, than when you walked in the door.

And those lectures were not designed to scare you off of that machine, or make you absolutely paranoid to use it. Because there are, relatively speaking, some pretty simple things, some pretty common sense things, easy things you can do to generally, you know, practice "safe computing," and not have to worry so much about what threats might be out there.

So just, rapid-fire, what are some of the threats or some of the concerns that you might have or face when using your computer on the Internet, as you know them or as maybe we discussed in previous weeks? What might you be worried about?

STUDENT: (inaudible response)

DAVID MALAN: Okay, so getting your computer infected with spyware, adware, call it whatever you want, but generally malware—malicious software. So what can you do?

Well, this article offers a number of bulleted suggestions, which we might go through rapid-fire, just to say what we or what I agree or disagree with. But, frankly—and this again is fairly subjective—I would say that the only types of things that you should really bother doing, or the only things you need to do to keep yourself relatively secure these days is have some kind of antispyware installed, and...

I mean, personally, I use Spybot. Why? Because it's free, it's relatively easy, it's updated pretty frequently, and that's sort of enough for me. Spyware can do any number of things, most of which tend to be bad. All right?

What about antivirus software, or anti-virus/-worm/-spyware/-software. The words used to describe these packages these days are becoming sort of overly specific or "underly" specific insofar as they do so many things.

So these shrink-wrapped boxes that you might buy these days might not fall perfectly into these categories, but it's the spirit of this software that's important.

What do I use personally for antivirus? AVG, which is the thing we link on the course's Website. Why? It's free, it's easy, it's pretty often updated, and it certainly does the job presumably as well as some of the more commercial products.

By contrast, Norton AntiVirus is popular. McAfee VirusScan is popular. Many of you might already have those on your computer because you bought it. Or a lot of computers—Dell, IBM—sometimes ship with this software these days.

The only reason I tend not to like the products that you tend to pay for is that they tend to do too much, and they try to do too much, and you get too many of these prompts. Even when we used Dawne's computer in class, there were a couple of popups that were coming up—not from spyware, but from the antispyware software. And, personally, it's a nuisance, I think, certainly for me or for… potentially for people.

And the reason that you get these kinds of popups is that software like that, typically that you pay for, tries to do too much in that it tries to infer even more about what's going on in your computer. And therefore, it tends to err on the side of paranoia.

The downside, or the flipside, of course, is that, if the computer thinks something bad might be going on but isn't sure, well, it punts. And it punts by giving you a popup and saying, "I don't know

if this is bad. It doesn't seem to be such a good thing. You now decide: Yes, or no? Do you want the action to proceed?" Frankly, if you get too many of these false positives or prompts, they just tend to get in the way, I think, of actual good computing. And so that's why we at least in the course recommend these free solutions, which do enough and at relatively low cost to you in terms of nuisance.

Other things you should do. What about wireless routers that a lot of you have? What do you need to know, or bear in mind when using these things in your home or office?

STUDENT: (inaudible response)

DAVID MALAN: So people could be hacking them and taking your information. Let's be more specific now. What does that mean—to be hacking them?

STUDENT: (inaudible response)

DAVID MALAN: Good. So the example we concluded with at the end of lecture 9, when Dan brought up the laptop, was that packet sniffing example, where while I was speaking, he was sniffing some of the packets that my computer was sending, the result of which was he had a list, a veritable list of everything I had done in the past few minutes. And we pulled out a couple of examples of, I think, the Websites that I had just happened to visit.

You are particularly vulnerable to wireless packet sniffing when you're using an access point that doesn't use any sort of security. So any time you go to some Internet café, or some airport, or some hotel, or, frankly, you just haven't gotten around to paying for Internet access in your apartment, and you see a list like this, not only with one access point but with half a dozen, all of which might have good signal strength, realize—and we unfortunately only have one that's proximal, that one on the wall—but in your apartments and homes you might see others. And certainly when you're traveling you might see others.

And ignoring for now the legality of tapping into someone else's paid-for Internet connection, if you just arbitrarily connect to one of these access points, obtained via DHCP, automatically an IP address through that access point, via which you can access the Internet through that person's connection, you have to beware that anyone else who is similarly connected to that access point, or even the owner of that access point, could be sniffing, logging, watching everything that goes through that access point. And you just have to bear that in mind.

For the most part, that's probably not such a huge deal, right? If you're pulling up weather.com, you probably don't really care if someone else knows that, even if you might feel a little violated, so far as privacy goes, that someone is bothering to do that.

(00:10:02)

But when it comes to sending instant messages, or emails, then you want to be a little more careful. But fortunately with email, in particular, there exist protections here.

When you check your email, depending on the server, you can sometimes visit your Webmail, for instance, not via http://, but via, for instance https:// .

And you might be able to do this these days with Hotmail and/or gmail and/or AOL mail. It depends on these services, whether or not they provide this feature. But if you see a URL beginning with https:// , be it a Website for ecommerce, or an email . . . Webmail-based system, well, then you can trust that, even if your transport layer is insecure—that is, your physical Internet connection might be vulnerable to attack—well, it doesn't matter, you know, for all intents and purposes, if your connection from point A to B is itself encrypted.

Alternatively, what can the owner of an access point do to provide some degree of protection for his own use of his own access point?

STUDENT: (inaudible response)

DAVID MALAN: Yeah, so you can turn on encryption for the access point itself, which means the connection between my laptop and the wireless antenna is secure. You have to bear in mind, especially if you're using someone else's Internet connection, someone could be packet sniffing from the wire, or anywhere else on the Internet, if they have physical access to that. But, again, thinking pragmatically and, you know, realistically, is someone really in the middle of the Internet going to care about what Websites you're visiting, or what instant messages you're sending? Probably not. So it's sort of a calculated risk that you're facing.

But certainly your whole traffic is not vulnerable to interception in this room. And that arguably is much more common when you sit down in an Internet café, or some other such public place today.

But you have to realize, when you have these wireless routers, you have two options: WPA sometimes, and WEP, and sometimes WPA2, which is a newer version of it. Which is these is the worse of the two?

STUDENT: (inaudible response)

DAVID MALAN: WEP, right? So it is fact that, given enough minutes and the right software, one of these guys right now could be sitting here figuring out the WEP key to some local access point, if it showed up in their neighborhood. Unfortunately, there are a lot of routers, sometimes the cheaper ones—though, as technology progresses, this is becoming less of a problem—that only support WEP. And, frankly, if you're worried about this in your home, worried about just neighbors being nosy, and so forth, you might as well spend the additional twenty bucks or fifty bucks, and choose a router that on the box specifically says it supports WPA, since if you use a pass phrase or a key that's long enough, right? If you use the key phrase "1-2-3-4," even if you're using WPA, doesn't take long for someone to figure that out. But if you use something that's ten characters long, or twenty—bearing in mind, you don't have to remember this thing, right, because you type it into your computer once and it remembers it—well, that's certainly a good and easy step for you to take.

Well, turning our attention momentarily, just to this article. Certainly read this at your leisure, if you would like. But just to debunk or validate some of the suggestions they make, on the very first page

of text, which is page 122, so the fix: "Keep your computer up-to-date" is a suggestion that *Money* ma... again, *Money* magazine makes to you.

What does this mean? Well, this means in Windows and Mac OS, turn on automatic updates. For the most part, for the typical user, that is a good thing. Because often the updates that Microsoft and Apple force down to your computer, when you turn on automatic updates, are in your best interests, so far as security goes.

By contrast, if you're a system administrator, running dozens of machines, servers, for whom uptime is critical, system administrators tend to be a little reluctant to let Microsoft and Apple dictate when their computers are going to be updated. So for someone who runs a business and a bunch of servers, those types of people probably don't want to turn these things on, only because sometimes improvements, things that are meant to be improvements, end up breaking things. And at least in some environments, that's not acceptable.

But for a typical user, I would say turn these on. And in fact, if you have something like Windows XP 2, Windows XP with Service Pack 2, it probably is on already. And how do you find out? You can go to the Security Control Panel, under Windows. And Mac OS has something similar.

Bullet 2: "Use security software... " Sure, I concur with that, based on what we said earlier.

"But don't depend on it." Useful! They can't really trust it. That's where I think you interject your own common sense. Don't click dangerous links, and don't click bogus attachments, or risky attachments.

This is a tricky one. Column 2: "Take away your PC's superpowers." What this means is that when you log into your PC usually, you have administrative privileges, which just means, for the most part, you can install software. Unfortunately, ideal as this idea is—and this means create a new user, called malan2, and only give that person user rights, not administrative rights—that just means that I can't do as much with that account as I might otherwise with an administrative account.

Unfortunately, a lot of Windows software might break or might not operate correctly if you don't have administrative privileges. So realistically I don't think this is a good option. So you'll hear it a lot, but in practice, it tends not to work well, at least for people who like to install software pretty often, or uninstall things.

Finally, "Get a router and lock it down." We touched upon that.

"Be careful at the coffee shop." There, too, just know—don't be scared, don't be paranoid, but just know what the risks are when you sit down and connect to a wireless access point. And realize that if you have one of those T-Mobile HotSpot accounts, via which you log in, remember that you're logging into a Web page when you do things like that at Starbucks. You are not logging in to the access points. Starbucks access point—correct me if I'm wrong—is not secure in the WPA or WEP sense. Which means, yes, you're logging into Starbucks, or T-Mobile, and saying, "I have an account with you. Therefore, let my Internet traffic go through." But you're not encrypting it in any way. So just again, be mindful of the distinction.

Finally, "Get smart and get real about passwords." Well, we had that discussion. Passwords like "1-2-3-4"—not so good. Reusing passwords on multiple sites—not so good. Post-It notes with passwords on your monitor—not ideal, but realistically, if you're not worried about the people in your home or office compromising your information, better to write the password down on a Post-It note than to forget it all together, or to choose one that's so easy to remember that it's easy for someone else to guess.

And finally, "Know how your computer watches you." Again, this is one of these concerns about cookies which tend to be a good thing, not so much a bad thing. But we had that discussion a while ago.

"Shred or smash" when you get rid of your computer, sell it, or whatever. You know from last lecture, how you can wipe the data, at least with free software. Rarely do you have to pay for any of these protections.

And finally, "Don't assume you are anonymous online." And you've seen that, too. Just as we knew exactly when the students were accessing the Website, we also, for the most part, knew what their IP address was, what their host's name was, what their browser was, who their ISP was. And if that same student subsequently sent us an email, bear in mind that we could correlate that kind of information with the headers in the email, and infer who it was that was visiting the Website. So just realize that that kind of information is out there.

So hopefully—long story short from Lectures 8 and 9—it's fine if you were spooked, but don't be spooked so much that you think that this is an untenable situation.

Whoo! With that said, questions? No? All right!

So let's go figure out how to make weatherchannel.com, and the like.

So we know what the Web is. It's a service. It's an application that runs on top of the Internet. What, then, is a Web server?

Won't work tonight to look at the slides, right? What's a Web server?

STUDENT: (inaudible response)

DAVID MALAN: Good, so it's just a server that holds information in the form of files. And those files are served by this server, this Web server, to users that visit it. The relationship that we'll be focusing on tonight is pretty much that between Web browser—like Netscape, IE, Firefox, whatever—and Web server.

Popular incarnations of Web servers, just so that you've heard it before, are something like IIS on Microsoft's platforms. So if you have a Web server running on a PC, it's probably Microsoft's Internet Information Services software.

If it's running on a Unix or Linux computer, it's probably something called HTTPd, which is made by Apache. Free software? It's probably fair to say that many, if not most Websites these days are run by Apache's software, or something similar in spirit, unless servers are themselves running on Windows, in which case you might instead use something like Microsoft's hardware.

The nice thing about the Web and about Website development: It doesn't really matter what server you're running on because, for the most part, any platform provides the basic set of features; for the most part—certainly at the level we'll be focusing on Website development in this class.

So a server is a computer, or at least it's a service being provided on a computer. When you log into a server, you have some kind of hard drive there, one or more hard drives, the root of which we often call "forward slash."

(writes on board): /

That denotes the root of a hard drive. Or we call it in the Windows world, what?

STUDENT: (inaudible response)

DAVID MALAN: Yeah, C: \

The stupid distinction, right? One goes left, one goes right, but they denote the same thing?

(00:20:00)

So just a single slash, usually in the world of Unix and Linux—which is what we'll focus on because it's more common, and it's also what we'll use for the final projects—this denotes the root of the hard drive.

Well, beneath the root you might have folders for a whole bunch of users. This might be Malan's directory; this might be Rei's directory, and so forth. And this might be called our "home directories" on the server. We log into this server, via, say, SSH, like you might have done before. And when you log in, you have access to files on the server, which might be laid out as follows.

Well, this is useful, so that you can store files, but you also want your files to be publicly accessible. So on a system like www.fas.harvard.edu, anything that you see on my Website on that page is actually in a directory called, by convention, public_html. All that is, is a folder in my account. And by definition of a Web server, anything that's in this folder will be publicly accessible. And presumably there's HTML pages; that is Web pages, files written in HTML in that folder, being made accessible to the world.

This is the setup you see on a shared server like Harvard's, the result of which is that you would visit my Web page here, as something like www.fas.harvard.edu/~malan. So if you've ever seen a Website that has a tilde and then some kind of username, it just means that that Website is on a shared server, with a bunch of other people, and this is Malan's personal directory.

So that's all fine and good, and I would show you this if FAS were actually working. But we're going to outsource everything tonight, which is just as well. Because this, arguably, is not the sexiest of URLs, right? Imagine telling people to visit me at www.fas.harvard.edu/~malan. No one's ever going to visit, right? Even if it's just malan.com, it's questionable whether anyone will visit. But certainly this is not memorable and it's not sexy. And this is why, in the Final Project this year for the first time, have we migrated to this third-party hosting approach, whereby each of you were invited to buy your own domain. And, based on the Final Project's Part I that have been coming in, most of you took that route, rather than the FAS route, though you were certainly welcome to just work on the local system, though not today.

Much more exciting, I think, this year than in past years is that you'll be able to and will be helped in setting up your own Website on the Web. And even though we saw in the Final Project proposals that some of you have sort of been wondering, "Okay, that might be all fine and good, but why? Who cares?" It's certainly the case that, more so than in years' past, is this question of "how do I make a Website?" becoming a much more common question. And fortunately it's becoming a much easier question to answer.

And so for those of you who are thinking, "All right, this might be fun to make a Website. But why would I ever want to keep running this Website, do realize—especially, ideally as an alum of this class—that if there comes a day where a buddy of yours says, "Hey, do you know how I can make a Website, or where I begin?" you'll at least exit this class, not only with savvy on other topics, but with an answer to that question.

And with the experience of setting one up yourself, you'll find that, one, it's not that hard; and, two, even if it's a little self-indulgent to put a lot of photos of you, and your friends, and your trips on the Web, even if no one ever visits it, I certainly feel and think personally there's this coolness factor with having myWebsite.com out there. And if nothing else, it should be a proud little moment for you to share with just some family and friends, and say, "Hey, look at me. I'm on the Internet. Could you even fathom me having done this a few months ago?" And if we can make a few feel that way, then hopefully our job will be done well.

So with that said, how does one go about making a Website in the real world? Right? This is all fine and good, and most of the setup that you'll do if you use your FAS accounts—which I'll refer you to the teaching fellows in Section and Workshop for what steps are different here—they're fundamentally the same. Just some of the names of the directories, and so forth will be different.

Where do you begin? How do you set up a Website? All right, well, fortunately, you can already answer the first question. How do you begin setting up a Website?

STUDENT: Take E-1.

DAVID MALAN: All right, this isn't that self-serving of us, right? There are other ways to answer that. So, how do you... what's the first step in setting up a Website? What do you obviously need to do?

STUDENT: You need a domain name.

DAVID MALAN: You need a domain name that's not taken. All right, how many of you found a domain name on the first try? Oh, that's pretty good, pretty good. I've sometimes spent hours on projects just trying to figure out a good name. How many of you spent more than just one try, finding a name? Wow! So that's impressive. So either, um... maybe I'm just not very good at this. That could be the explanation.

All right, so you've got your domain name. Well, what does that mean? Well, we sent you to GoDaddy.com, if only because, one, we've used it ourselves; and, two, it does tend to be very popular, even though hopefully you realize how ridiculous all of the "buy, buy, buy" splashy banners were on that Website. I mean, they make it hard to find how to just check out, without paying more than $8.99. But again, that's sort of a lesson unto itself, navigating what's daresay a poorly designed Website from a user's perspective, but a brilliant design from someone who's trying to upsell you's perspective.

So with that said, you check out. You now own malan.com, for instance. And what does that mean, exactly? Well, what you have just done is registered with an authorized authority—that being GoDaddy—your domain name, for a year. You effectively have ownership of that for at least a year, assuming you paid for just one year.

What GoDaddy then does is registers that name with the authorities that run the Internet, so that effectively it is reserved for you. Right? There has to be some kind of agreement in the world. Otherwise this whole system wouldn't work. If someone from this country could buy malan.com, someone from another country could buy malan.com, the whole setup just doesn't work.

So there is something going on behind the scenes, when you buy that domain name, such that the world effectively is being informed that you now own that name. And no one else—even if they go through some other registrar—can buy that domain name.

Well, what then is step two? Well, you own malan.com, or some such domain name, but that's not a Website in and of itself. That's just the name. And that name might be used, yes, for a Website, but it can also be used for email, or for FTP, or SFTP, or any other service you might want to provide.

We in the course will do Web and email, since they're the two most popular. But you can do anything with that hostname that you can do with the Internet.

If you want to set up a Website, what, based on what you might have inferred from the project spec, do you next need to go find for yourself?

STUDENT: (inaudible response)

DAVID MALAN: So you need a host. And this is, in and of itself, is kind of an overwhelming thing, which we've tried to save you the headache of. If you google the term "Web host"—which is the common moniker given to people who give you the ability to map your domain name to a physical server like this, so you can actually post your content to the world—if we search for "Web host," frankly it's just overwhelming. Like, where do you begin? And this is just the first page of results. They'll go on forever.

You might have seen that GoDaddy itself offers Web hosts, and they might be fine. DreamHost.com is the host that we're currently using for a number of reasons, not the least of which is the fact that with our one account can we host an unlimited number of domains.

Now, this is kind of the exciting thing, exciting in a geeky way. Because of all of the competition out there for Web hosting, and because of the relative cheapness of providing Web hosting services today, you get ridiculous deals.

So, for instance, this is DreamHost.com, and you yourselves will not have to visit this Website, per se. But just to give you a sense of where we began—so that you could repeat these steps if you signed up for a host not through us, but on your own—we signed up for, I think, a one-year, or a monthly plan, which I think… We have the "code-monster plan," which... Oh, as you would see, it's now cheaper than we're currently paying. That's wonderful. Just like that computer of mine.

If we scroll down. So we're in column three here, level three. So it's, you know, $15.95... $19.95 a month; twenty bucks—not horrible; a lot cheaper than your Internet service probably. What are we getting in our account?

Well, we have a 97-day money-back guarantee, if we're very disappointed in your submissions. Kidding! We have 24-hour technical support, and so forth. But what's more interesting: 400 GB of storage space. So even though some of you might have thought it was kind of neat that we offered you in the spec a GB of storage for yourself, we're actually keeping most of it for ourselves, right? So, we'll give you more, if you want it. But it's sort of ridiculous how much you do get at your disposal if you pay just this $20-a-month rate.

Automatically increasing bandwidth: As though that weren't good enough, every week they're apparently throwing 2 GB more at us.

So monthly bandwidth at signup: This too is good, especially when our podcast was formerly hosted in this account—4 terabytes per month. A terabyte (TB) is how many gigabytes?

STUDENT: (inaudible response)

DAVID MALAN: A thousand, or 1,024. And a gigabyte is how many megabytes?

STUDENT: (inaudible response)

DAVID MALAN: 1,024, and a megabyte is how many bytes... or kilobytes? I'm not following the pattern here.

1,024—and a kilobyte is how many bytes? 1,024—so we're talking about a trillion bytes per month; 4 trillion, in fact.

We don't foresee us going over this, even with all of these Websites being hosted.

And, long story short, you'll see a whole bunch of other services offered; features, some of which are more marketing speak than they are actual features. But things to bear in mind when you're signing up for your own Web host is what kind of email support might I get? Sort of closely coupled with the idea of having a Website is having email addresses in that same domain.

(00:30:10)

So the options that you have when setting up email are most popularly something called POP, if you'll recall, and IMAP. Think way back to Lectures 4 and 5. What was a distinction between POP and IMAP, as you might recall from your reading, or from class, or experience?

Sort of a useful thing to keep in mind.

STUDENT: (inaudible response)

DAVID MALAN: POP—it doesn't quite forward. But if you think of the acronym as implying that it pops mail off the server, it does do that. When you have a POP-based email account, david@malan.com, what that means is when you check your mail, your mail is being copied from the server down to your local client, and then, depending on how you've configured your email client, it's sometimes or always deleted from the server subsequently. The result of which is you have your email on your local computer now. But if you are one wont to travel or check your email from home and work, if you subsequently try to access your email from your work computer, what will you see in your inbox?

Nothing. And that, in and of itself—especially as people become more transient in their Internet usage—that, in and of itself, can certainly be a nuisance.

My mom for a while only could get POP email access from her Internet Service Provider, and this was just a common frustration. Because if she'd go visit family sometimes out of state, she would then have no access to her email because it was on her local client.

One obvious solution is don't let it delete from the server, right? Just leave everything on the server and on your client. What becomes problematic in that case?

STUDENT: (inaudible response)

DAVID MALAN: Sorry?

STUDENT: (inaudible response)

DAVID MALAN: Double. You have this inconsistency problem. Plus it becomes annoying, if nothing else, that you have all these old emails sort of intermingled with emails you might still care about. Two, you won't get from a typical ISP these ridiculous amounts of storage—yes, with your gmail account; no, with your Comcast account. The result of which is that you'll be forced, effectively, to eventually start deleting emails, and attachments, big emails, and so forth. And that too can just be a nuisance.

So by contrast, IMAP still lets you access email but it maintains synchronicity between your client and the server; which is to say, if you check your email from home, not only will the emails be downloaded from your computer, they'll remain on the server until you delete them from your client. The moment you hit Ctrl-D on your client, a message will be sent to the server, saying also delete it from the server. And in this way do you get synchronicity between the two. The result of which is, if you then check your email elsewhere, either with Outlook or maybe Webmail from your ISP, what you see in your inbox should be exactly what's sitting on home as well. And if you make changes while at the office to your email account, and then log back in at work, and pull up Outlook, or whatever program, the moment it connects to the server, even though briefly you'll see old messages, they should disappear, assuming you've deleted them elsewhere. And it does automatic synchronization.

Long story short, if you have the option for IMAP, it certainly tends to be better these days. And I would certainly, if choosing to host a Website, look for IMAP as a feature. It's not uncommon but sometimes it's not in fact offered.

So, beyond that, the rest is sort of icing on the cake. If you're a real Web developer, and you really want something technically specific, the rest of these options might be of particular import to you. But for the most part, you probably care about how much space you get, how much it's going to cost you, what kind of email access you have, and that's about it.

Well, what do you get once you sign up for an account? If I scrolled down here and said, "Sign up now," as we, the course already have, you would receive some kind of account from DreamHost.com, and you would receive a username and password. You could then log into DreamHost's account, and you can specify, "I want to now start setting up malan.com. And the neat thing about DreamHost, for instance—and we don't get any kind of kickbacks from DreamHost, right? We're paying them. So, lest this come across as an advert, like our Coke, and everything else; American Express, earlier tonight. We don't get anything out of this. They're just good to use. You can host an unlimited number of domains with DreamHost, which is sort of crazy, which is how we're able to host thirty-some-odd Websites using this account for the purposes of the course.

Well, what we're doing, and what these guys are currently doing, hopefully, is creating your DreamHost accounts, based on your submissions of Final Project: Part I. What you'll receive by email from us shortly is exactly what you see on one of tonight's handouts, which is about Final Project: Part II, which you should have in your hands? Yes? No?

STUDENT: (inaudible response)

DAVID MALAN: Yes? Yes. No! No? Yes? No.

STUDENT: (inaudible response)

DAVID MALAN: That's Exam 2. We should have one Final Project: Part II?

STUDENT: (inaudible response)

DAVID MALAN: Surprise! Okay. Available on the course's Website, though not tonight, is the second part of the project. Sorry, it didn't seem to come back from the printers. So, that's okay, it's on the Web, in theory.

What you'll see on that handout, when you're finally allowed to see it, is a bunch of blank fields. And on the Final Project: Part II, which, dare I try to access it now? Let me pull it up for just one second, because this will be helpful as a walkthrough for your own Final Projects.

So I am just using Secure FTP to access the course's account via a protocol, a service—SFTP, which for whatever reason, is working today, even though the Website is not. I'm going into our Final Project directory, grabbing the spec, putting it on the desktop, and opening it here. And what you should have in your hands, had it come back, is the following.

So the top part of this document specifies these fields. So what you're going to get from us shortly is an email, providing you with a reminder of what domain name you bought, that should not be new information; the email address that you've requested we set up for you; and a password that we're going to set up for you. Mind you, as we say in a footnote here, emailing around your passwords is in stark opposition to the recommendations we made two weeks ago. However, with a class of fifty students, the fact of the matter is, it's much easier to email you all your passwords than going around whispering it in your ears, or calling you on the phone with a password. We are going to hope that there's no bad guys sitting between you and us when we send these emails. But realize this is not best practice. Though, frankly, the odds of someone being really interested in stealing the information you're about to make publicly accessible on the Web? Probably not such a concern. But realize the hypocrisy in this.

So we're going to give you that information. We're going to give you this: seamus.dreamhost.com is the server on which all of your Websites will be hosted. More on that in just a moment. So what that means... when we signed up with DreamHost, we were given an account on a server called seamus. We are similarly going to create other accounts for you, such that when you log into seamus.dreamhost.com—using your FTP username and FTP password, which we'll also provide to you in this email—when you connect via SFTP, you will see a root directory, as though that's the server. But the nice thing is about this setup, is that what you'll then see is three directories—two of which are irrelevant; one of which is relevant, which is simply going to be called domain.tld, where domain.tld is whatever you bought.

What that means is that when you log in via FTP—which, if you haven't used it before, it's, one, very easy to use; two, we'll do it in Section and/or Workshop in the coming week—you will have this folder. And anything you put in this folder will become accessible on the Web; specifically it will become accessible at www.domain.tld, or just domain.tld. It's literally going to be as simple as that.

The reason that you get this illusion, as though you were the only domain on the server, is because DreamHost, like most every other Web host out there, supports what's called "virtual hosting," which essentially means they use one server—for instance, seamus.dreamhost.com—to store a whole bunch of Websites. But they configure it in a way that creates the illusion of each of these

domains essentially being its own Web server, even though it's being cohosted with a whole bunch of other Websites.

The upside of this is that relatively cheaply can DreamHost host a whole bunch of Websites. The downside, potentially, is if you're account is on the same box as a really popular Website, whatever it happens to be, well, your Website might be slow. And so what you have to realize is, even though we're only paying twenty bucks a month, which is pretty cheap, you can pay a dollar a year with some Web hosts these days, and get your own Web host, with, like, a GB of storage and unlimited email addresses, and so forth. But you really get what you pay for. Right?

I was doing this for a year with a site, just personally, that frankly, if it went down, the world would not notice. And that was fine. It was just a test Website I was playing for, but I got what I paid for. Right? Within two months, I had lost my dollar and all of the content I had uploaded to the server.

But if you do this on your own, certainly beyond this course, just so you have some other options in mind, DreamHost is fine. Other ones that personally I've used, if you want to keep them in mind: thehostgroup.com has been good to me. A very popular one is called pair.net, if I'm getting the domain name correctly. That's more of a high-volume-type Website, way more than someone, like an individual person, might likely need. And there are a whole bunch... Oh, there's another one I've used: e-rice.net, among the weirder ones, but it's like ten bucks a year.

(00:40:06)

So if again you want an account that you don't really care about but it's useful to have for maybe a secondary email account, or some basic Web space, those guys have been good to me as well. And I point these out only because, there are so many damn options on the Web, that finding a Web host is sort of a challenge sometimes unto itself. And the best solution to that problem usually is asking around for recommendations—people at work, friends, and so forth.

All right, so, you will get—and if you do this outside of this course, you will similarly get all of that kind of information, login information, after you've bought your domain.

So what do you have to do to tell the world that subsequently malan.com, or domain.tld, needs to point very specifically to that server? Right? Because if you just bought your domain at GoDaddy, what you'll see if you pull up the Website is the following.

Would anyone mind revealing your domain name, so we can pull it up?

STUDENT: (inaudible response)

DAVID MALAN: So if you just bought your domain, and you might have done this already, you'll see a page like this. This means your domain is currently "parked" at GoDaddy.com. That just means you don't really have a Web-hosting account with GoDaddy. They just, by default, have chosen to proclaim to the world that this domain is owned by someone, and it's parked at GoDaddy. The alternative would just be to display an error message, or nothing at all. This, in and of itself, is not particularly useful, except it's a wonderful ad now for GoDaddy's services.

So that's not useful, and we're not going to put our files on GoDaddy. Rather we're going to put them on DreamHost. So what do you need to do at GoDaddy?

Well, the only thing you'll need to do—and the spec, too, will remind you of this process; and this, too, is exactly what you would do, not just for E-1, not just for your Final Project, but for Website development and hosting in general—is the following.

You would log into whatever site you'd bought the domain name from. You're going to type your username or customer number, which I can't remember, so I wrote it on my laptop there. And I'm going to paste this in here. And I won't tell you our password. But I'm going to log in to GoDaddy.com. And again the spec, which you'll soon have in your hands, will walk you through this. You'll go up to domains, my domains. And this, of course, will vary, based on registrar. But, frankly, there's little reason not to just keep using the same registrar with which you're already familiar. It's hard to beat ten dollars a year. You can, but it's not going to save you all that much.

So malan.com, unfortunately, was taken, but I did go out and get us malanrouge.com. This, of course, is a pun. I was not so clever to come up with this. An office mate of mine, when we were first choosing our desks, four years ago, when I returned to grad school, plastered a sign over my desk, saying this is Malan Rouge's desk. So I thought we'd go with the domain name.

And, notice, I splurged a little bit, right? I wasn't happy with just a dot.com. And for vanity's sake or also just discussion's sake, officially, I also bought the dot.net and the dot.org. Maybe an easy question, but why? Yeah?

STUDENT: (inaudible response)

DAVID MALAN: Good. So, one, if people mistakenly type the wrong one, at least you can map that to your own domain name. And, two, it's not uncommon for people to buy bogus domain names, or domain names with slight misspellings, or different TLDs just to make fun of some company, or just make money off of sites where, you pull up a slight misspelling of very popular Websites, you'll usually get some kind of advertisements pushed at you. If nothing else, this slightly discourages it. Though, frankly, given... You saw at GoDaddy how many TLDs are available to you. You can very easily spend way more money than you should, getting all of the TLDs that realistically no one is accidentally going to type "malanrouge.info," right? So I didn't go that far in this case.

But I did go this route for tonight so that we could show you, if you did go this route yourself, well, how you can map all of those domains to just one.

And the simplest way to set up that, as well as the simple task you must achieve to map your domain name to your server, is the following. When I pulled up my account and went to my domains, I see this page. And you too will soon be doing this at home. You will then simply click on the domain name that you want to modify. I'll go with the dot-com and make that my primary domain. You'll see some scary-looking information, or just a lot of details, almost all of which you can ignore.

Because the only thing you'll need to do is the following. There's an icon up there, called "Nameservers," and you're going to click this. You're then going to see a form that looks like this. And the only thing you have to do henceforth at GoDaddy, until you want to move your Website elsewhere or renew your domain name, is fill in these top three boxes. To be very specifically: "NS1.DREAMHOST.COM", "NS2.DREAMHOST.COM", and "NS3. .DREAMHOST.COM".

"NS" stands for "Nameserver." What this is doing is the following. It is informing the world, the moment I click "Okay," that, yeah, I bought malanrouge.com from GoDaddy, but that's uninteresting. What it's telling the world is that henceforth, any requests by some random Internet user's laptop or desktop for malanrouge.com should be referred to one of these servers at DreamHost.com. Because guess what DreamHost is going to do for us?

Well, when you visit www.malanrouge.com with your browser, that's not sufficient to getting from point A to B. What does your computer need in order to visit what is malanrouge.com? Think back to our envelope passing, a while back.

STUDENT: (inaudible response)

DAVID MALAN: The server?

STUDENT: The address.

DAVID MALAN: The address. What kind of address?

STUDENT: IP.

DAVID MALAN: The IP address, right? When we talked about routers—those big computers that have tables that decide whether the packet goes this way or that way—right, they had tables that essentially relied on the use of IP addresses, saying anything that starts with 140.247 should go that way. Anything that starts with a different number should go that way.

Well, your request for malanrouge.com needs to get translated to an IP address. A DNS server, aka, a name server, is what does this. The sole purpose in life, for the most part, of a DNS server is to answer questions of the form "What is the IP address of malanrouge.com?"; "What is the IP address of weatherchannel.com?"; "What is the IP address of CNN.com?"

And can also reverse the answer. You can say, "Here's an IP address. What Web server does this belong to?"

So that's useful, because it means you, the user, never have to know what's going on underneath the hood with IP addresses, with routers, with the Internet. And it's what ultimately makes the Internet work as seamlessly as it tends to for typical users.

So the moment I click "Okay," I am now telling the world that any requests—be it via email; or Web; or instant messaging, if I'm running my own instant messaging service—any such requests for IP address should be referred now to DreamHost. And DreamHost is going to tell the world what

malanrouge.com's IP address is. And it's specifically going to tell the world that the IP address of malanrouge.com is identical to that for seamus.dreamhost.com.

So this is what it means to be virtually hosted. The official name of the server on which my Website will be hosted will be seamus.dreamhost.com. Could be anything else. Could just be a specific IP address. But for now it's seamus.dreamhost.com. But because of DNS and because of virtual Web hosting, as it's called, multiple host names—malanrouge.com, reidiaz.com, dan.com—can all point to the same server. And it's because DNS servers ultimately tell the world what IP address relates to a domain name.

So if that DNS server at DreamHost.com, for everyone in this room, as of tonight, starts answering questions of the form, "What is David's IP address? What is Dan's IP address? What is Chris' IP address?" And gives the same answer to every query, and specifically gives the IP address of seamus.dreamhost.com. Then all of our requests for Web pages for students in this class will be mapped to the same server. And that server will then determine, based on what name was requested, whose directory the request should be sent to.

So let's put this... let's validate this. So I'm going to pull up a command prompt in Windows here. I will make this slightly bigger, and change our font size to something a little larger.

And what I am going to type is "nslookup seamus.dreamhost.com," enter. This program, which is available on many computers, translates seamus.dreamhost.com to its IP address. How does it do that? It asks the nearest DNS server, "What is the IP address of seamus.dreamhost.com?" That's all.

And in fact, it looks like seamus.dreamhost.com has the IP address of 208.113.148.29. Well, wait a minute, what if I similarly say, "All right, nslookup malanrouge.com?" If this demo works, what should I see?

STUDENT: (inaudible response)

(00:50:04)

DAVID MALAN: Okay, so, why might this be? So let's extract a lesson from this. Servers, turns out, can sometimes have multiple IP addresses, which is exactly where we're going with this. So this is why... That's always great when that works.

So this is perfectly okay, because even though my content will in fact be hosted on seamus.dreamhost.com, it's possible for, one, a server to have multiple IP addresses that both physically refer to the same machine. What does that mean? It usually means they have two Ethernet cards, sometimes for bandwidth reasons, sometimes for redundancy reasons. It depends. It could also mean that, just like FAS allows you to access your files from multiple different login machines, it could just be that my files—even though dreamhost has been telling us to go to seamus.dreamhost.com, they're might be other servers that provide me with access to my files.

The short answer is it's unclear, from what just happened, which scenario it is. But, when we do finally pull up malanrouge.com, it will be resolved to that IP address, which ultimately will map to,

somehow or other, the files that I put on to seamus.dreamhost.com. And my Website should come up.

Well, let's confirm this. I am going to do the following. I am going to, with our FTP program—which on the PC is SecureFX. I'm going to connect to seamus.dreamhost.com as malan, and then with my password, which I can't tell you. And notice, that what I have here is a layout that looks very similar to this. And I said there are three folders, but I said, ignore the other two. Well, there are the other two. You'll have this too. One is called logs, one is called maildir. Maildir, as you might guess, is where email is going to end up.

But for tonight, we're just going to go into this directory. And it looks like there's three files in there. For now let's ignore that. And let's instead open up a local copy.

How do you make a Website? Well, quite simply I am going to open this file, and this will be a little teaser before we take break.

Every Web page you make in this class will start with this scary sequence of three lines.

Note, on screen: a Document Type Declaration, or DTD

Okay, personally, I never remember how to type these lines, so I always copy and paste them from a previous file. So don't be daunted by this. And this is actually a little small. Let me very quickly make this a little larger for us.

Ah, let's make this... let's see how we do this. Document Classes/ HTML

What you will see... There we go. We recommend in the Final Project spec that you not just use programs like Notepad, or similarly cheap and dumb programs, because it'll just be black and white text. The nice thing about a program like I'm using, TextPad, which we refer to in the spec, is that it does something called syntax highlighting, which means all of the interesting parts of your documents will be highlighted automatically in a different color.

Well, how do you make a Web page? Here is your first introduction to making a Web page.

Every Web page starts with that.

Note, on screen: <html>

And we'll take the hood off of this example after break. But I'm going to say This is Malan Rouge.
And a <title>
I'm going say <head>
I'm going to say <body bgcolor="white">
Then I'm going to say <img src="malanrou . . .
Oh, I'll just say hello. Welcome to Malan Rouge!
</body>
I'm going to delete some blank lines.

Note, on screen:

```
<html>
        <head>
          <title>This is Malan Rouge!</title>
        </head>
        <body bgcolor="white">
        Welcome to Malan Rouge!
        </body>
</html>
```

You don't have to be writing this down. Don't worry. And I'm going to go to the File menu. I'm going to save it. And I'm going to call it test.html. I'm going to hit Enter. I'm going to go back to this FTP program. And quite simply, all I'm going to do—and you shouldn't be seeing that in advance—is I'm going to drag test.html to my malanrouge.com directory. And notice that it appears at right. Now it's over here, which means it's on the server.

So now I'm going to go up here and go to malanrouge.com/test.html. We'll try a little finger-crossing, since it worked so well last time.

When I cross my fingers and hit Enter, what should I now see?

STUDENT: (inaudible response)

DAVID MALAN: What I just typed. Whoo! All right. One for one tonight.

So, I have just made a Web page on the Internet. Not that hard. Also not that dramatic. So just before we take our break here, let's suppose that I know a few more HTML skills, which, within an hour's time, you too will.

What do I need to do to enhance this thing? Well, let's forget about the text. And let's instead use an image that I spent far too much time on Photoshop with. And insert this malanrouge.jpg, whatever that is.

Note, on screen:

```
<html>
        <head>
          <title>This is Malan Rouge!</title>
        </head>
        <body bgcolor="white">
          <img src="malanrouge.jpg" />
        </body>
</html>
```

Let's go ahead and refresh it. And I'm going to reupload the file to the server, right? Because I've just changed the local file. If I refresh this, nothing material has happened because I haven't changed what's on the server. So bear that in mind.

FTP. I'm going to take this file, test.html, move it over there. And now, prepare to be... oh, you won't be amazed yet. I'm going to change the color to be black instead of white.

Note, on screen:

```
<html>
        <head>
          <title>This is Malan Rouge!</title>
        </head>
        <body bgcolor="black">
          <img src="malanrouge.jpg" />
        </body>
</html>
```

It will make the Website look so much better. Going to reupload that, just by dragging. That's all it takes.

I'm going to go back to the Website. Just like a Web page you might visit. Just hit Refresh, and it will regrab the newest copy of the Web page and, drumroll... Wow! Look at that Photoshop work! Look what I did for my latest problem set!

So this is all fine and good, but who the heck wants to visit malanrouge.com/test.html? How do you make it the default page?

Well, it turns out that all you have to do is name that same file index.html, which on most Web servers is the default page that a Web browser will retrieve if you don't specify a file name.

So if I want my so-called home page to be that same thing, I just have to rename it index.html. I'm going to center the image this time.

And now notice that, one, test.html is gone. If this is a good lesson... If you see things like this, "Not Found"—fortunately, this error message, unlike some, kind of tells you what's wrong: The file's not there. You mistyped it, you had a space in the file name that you didn't actually type. You have different capitalization. These names are case-sensitive.

Well, now, though, because I named it index.html, I can just go to malanrouge.com, and, drumroll, there is our first Website. Nothing more humble than that, right?

On that note, why don't we take a five-minute break, and when we resume, we will take the hood off of weatherchannel.com, so to speak, and show you how to make things far more interesting than that.

Hour 2
(00:56:45)

All right. So, in last hour's episode, we, one, bought a domain name; we, two, configured the registrar of that domain name to inform the world that the Nameservers that should resolve IP address queries for that domain name are at dreamhost—specifically NS1, NS2, and NS3.

We then set up our account with DreamHost.com, such that I logged into my account. Then I had gotten this username and password from my Web host. And I simply started uploading files to the folder that my Web host had told me to put all of my content in.

So why did this not work when we did this whole IP address trick earlier? In other words, how is it that virtual hosting works, and yet did not seem to work in our demonstration, a moment ago?

Well, we've solved the problem. So when we earlier asked for the IP address of seamus.dreamhost.com, we got back this number that ended in ".29".

It turns out that seamus.dreamhost.com appears to be the FTP server for dreamhost. And by FTP server we mean File Transfer Protocol, which means this server's job in life is to allow people to upload and download files via an FTP client. And that's better than a Web page, which really only lets you download files in one direction, and it doesn't let you do this drag and drop, and so forth. So that's useful for us, so we can get our content there and back.

However, the Web server that is virtually hosting malanrouge.com, and all of your Websites, henceforth, is a different machine all together, or at least a machine with a different IP address. And that IP address we saw was the one ending in ".78".

The reason I just came to this conclusion was by the following test. I simply brought up a Web browser, and instead of visiting malanrouge.com, I decided, you know what? Let me visit the IP address that malanrouge.com actually is. So that is 208.113.148.78, Enter. I got this message

> Note, on screen: "Site Temporarily Unavailable".

But malanrouge.com seems to be accessible. But that's okay. Because the way we have our server set up is that my Website is being virtually hosted, which means the only way to reach it is by its domain name, not by its IP address. The reason being, if multiple Websites are being hosted on this server virtually, so to speak, well, which one should I expect to pop up, anyway? And so the way DreamHost has things configured, right? If they were GoDaddy, they might give a bunch of ads to us. But they just have this error message, which just means, "What Website do you even want, really?" It's unclear.

So that's to be expected. Now suppose I pull up, not just the IP address of malanrouge.com, but this other one of seamus.dreamhost.com, which recall, ended in ".29".

Well, what led me to the conclusion I offered a moment ago was this: Nothing's happening. Right? It looks like... And now it's just timing out. It's not doing anything, my browser, which seems to

suggest that the IP address that belongs to seamus.dreamhost.com is not in fact a Web server. If it were, that should work. We should at least get something back.

So the hypothesis I'll offer is that seamus.dreamhost.com is an FTP server, via which we and you will be able to upload and download your files, but only via which you'll be able to upload and download your files.

(01:00:06)

To access your Website, obviously, you'll visit yourdomainname.com. But your Website physically is going to be hosted on a different machine, whose name we don't care about, because all of that happens behind the scenes, seamlessly for you.

But what is important to note, is that even though seamus.dreamhost.com's IP address refers to one server, an FTP server, and the IP address that apparently refers to my Website is a different number or maybe a different server, they both have access to the same hard drives, or to the same storage space. Because obviously otherwise this would not work.

And this is a common practice to be able to log into multiple machines, via maybe different protocols or services, but still have access to the same content. And that's simply the way their system administrators have apparently set things up.

Henceforth, it's not interesting to you what your IP address is, certainly when you're hosting a Website, because the DNS servers take care of all of that for you. Neither you nor your visitors need to know.

So how do you go about writing something… This is like a Web page written in Photoshop, right? I haven't really done anything interesting here. I've just plastered a big graphic on my Web page.

How can I actually start doing things with text, and links, and mailto links, and other sorts of things? Well, let's start at the beginning.

Here's where we began, which was with this XHTML, which I just whipped up pretty quickly, right? Even I copy-paste the first three lines because I never remember them, and that's fine.

Note, on screen:

```
<html>
        <head>
          <title>This is Malan Rouge!</title>
        </head>
        <body bgcolor="black">
          <img src="malanrouge.jpg" />
        </body>
</html>
```

In this class we'll be writing XHTML, which stands for "Extensible HyperText Markup Language." Don't need to write that down because you don't need to repeat it, but it's also on your Jargon sheets. But if you can remember it, great. But all it means is that XHTML is a markup language. And as that kind of implies, it's a language. It's not a programming language. You can't make software with it. But you can mark up content and make things look bold, or big, or small, or blue, or red, black, or white, because you can mark up text. And you can include images, and also sound, and videos these days. And you do this by writing XHTML.

As an aside, many Websites, maybe most Websites are written in a looser, less robust language called HTML, which just doesn't have some of the same constraints that we're imposing. But increasingly know that Websites are beginning to be written more in XHTML because it is a superior language from the computer's perspective.

So for now just think that XHTML and HTML are pretty much the same. Except with HTML, you get to cut a lot of corners, and a lot of corners that, frankly, sometimes result in Web pages displaying content differently, because if you're being sort of lax with the rules, a browser has to make a decision as to what you meant. And sometimes Firefox, say, will think you meant something different than Internet Explorer meant.

But even besides that, you'll see, as you develop your projects, dealing with browser differences is a pain. It is a nightmare, a headache. And sadly, it hasn't gotten hugely better over the years, where Microsoft will decide to implement some feature in one way, and Firefox will interpret that feature in another way. The result of which is your Web page looks different on different browsers.

Which means part of the development process—certainly for corporate sites, or sites where you care about the user's experience—you end up developing them with Internet Explorer open in one window; Firefox in another; a Macintosh computer on the other to run Safari. It really is an involved process, but it's really just a quality-assurance process. And usually it's easy to fix inconsistencies.

But you have to beware that what looks right on your screen might not look right on someone else's screen. And that's important, certainly, for business sites.

So what is all this stuff? Well, for now, just assume that the top few lines, four lines of any XHTML document you write needs to look like that.

Note, on screen: a DTD

And don't worry about what it is. Just copy-paste it. The interesting stuff, and the stuff that you'll get to spend your time on are these things.

Note, on screen:

```
<html>
        <head>
          <title>This is Malan Rouge!</title>
        </head>
        <body bgcolor="black">
          <img src="malanrouge.jpg" />
        </body>
</html>
```

So what immediately jumps out at you at what I've written here? Describe it in sort of high-level terms. What's interesting about those several lines of code at the bottom?

STUDENT: (inaudible response)

DAVID MALAN: Everything I start I end. Example?

STUDENT: (inaudible response)

DAVID MALAN: Title. I seem to have one instance of <title>, and then another, but the second is slightly different. What's different?

STUDENT: A backslash.

DAVID MALAN: Uh, close. We'll get the jargon right.

STUDENT: (inaudible response)

DAVID MALAN: Forward slash. Minor point. And maybe a religious debate more than an important debate. But forward slash, same word.

So each of these things that begins with an open bracket, <, as we'll call it, and ends with a closed bracket, >, is called a tag.

So when you write XHTML or HTML, it's all about writing, not only your content—like the actual words the user sees—but marking it up with tags. And tags are not data so much as they are what a CS person would call "metadata," or "semantic information," whereby it has meaning to the computer, but the user doesn't need to know that that tag is behind the scenes.

So nothing in these open brackets and closed brackets actually should appear to the user, unless you've made a typo, in which case it might actually appear erroneously.

Any Web page you write—besides starting with this crazy stuff (the DTD)—must start with open bracket, html, closed bracket: <html>. It is a paradox that when you write XHTML, you must start your document with <html>. So be it!

Any document you write must, therefore, end with open bracket, forward slash, html, close bracket: </html>. And this is sort of an easy trick to get used to. And this really is the difference between XHTML and HTML. With HTML, you don't necessarily have to close your tags, so to speak. You can open them but you never have to get around to closing them, the result of which is it's much harder for a computer to parse, that is, read, the Web page. So we preach the better of the two, XHTML.

So with that said, any tag you open, you've got to eventually close, and you've got to open and close it in sort of opposite order. So the first tag you open is going to be the last tag that you close. So everything gets nicely balanced in that regard. It's symmetric.

The next tag you'll write, after <html>, is the head tag: <head>. And inside the <head>, so to speak, goes your title tag: <title>.

And if you decide to write more advanced Web pages that might have little scripts and programs written in JavaScript, sometimes in the head tag, will you also have mini-programs, using another type of tag.

But for now, know that you simply must have a title tag: <title>.

Below that, we're all done, so we close the head tag, so to speak: </head>. And then we open the body tag: <body bgcolor="black">.

We type whatever our content is for the body of the Web page. Recall that originally it was something like, "Welcome to Malan Rouge," I think, something like that. Close body: </body>. Close html: </html>. Now we have a Web page.

Note, on screen:

```
<html>
        <head>
          <title>Welcome to Malan Rouge!</title>
        </head>
        <body bgcolor="black">
          <img src="malanrouge.jpg" />
        </body>
</html>
```

The two things, therefore, that every Web page must have is a <head> and a <body>. Where is this <head> appearing? Stuff in the <head>, specifically the title tag appears on your Web page? Did you notice?

STUDENT: At the top.

DAVID MALAN: The very top in the so-called title bar. So notice it has "Malan Rouge!" That's the title. And the only thing you would put in the title tag is the... Oh, I see, it's a different version here that I'm playing with. So let me... Lest we wonder about the inconsistency.

So the file that's on the server is this one, whose title is "Malan Rouge!" and that's what appears in the browser window.

That's also what appears, notice, down here in the taskbar on, say, a Windows PC. That's all.

The body is where all the juicy stuff really happens. So let's tease this apart. I'm going to restore it to where it was a moment ago.

So our original Web page was white. And I simply said, "Welcome to Malan Rouge!"

Let's go ahead and upload this thing to the server. Uploading is as simple as dragging and dropping. And we'll cover this in Section and Workshop, if you are not so familiar.

Let's refresh it, and here's the Web page... Oh, that's that not it. Let's upload index.html. It worked that time. And here we go. We're back to our very boring Web page.

So what can we begin to do to control the behavior of a Web page? All right, well, I deleted some of the interesting stuff, but I left one remnant: bgcolor. So, whereas the name of that tag is the body tag: <body> , tags in XHTML can also have what are called "attributes." An attribute is simply something that modifies the behavior of the tag.

So bgcolor, as you might guess, denotes what?

STUDENT: (inaudible response)

DAVID MALAN: Background color. What you'll notice in XHMTL is there are lots of nicknames, really, for things, because it's much quicker to type "bgcolor" than "background color." So you just have to get used to some of these nicknames.

bgcolor is an attribute, which means that it is something of the form name = "value". The name of attribute just tells the browser what behavior modification it needs to impose. And the value specifies how it should control that behavior.

Put into a much more accessible context, <body bgcolor = "white"> clearly means "make the body of the Web page have a background that is white.

So to convince you of this distinction, what if we do, well, black: <body bgcolor = "black"> ? Right? This is not magical at all. I'm just going to go back to the files, and I'm going to upload index.html again; open my browser; refresh. Well, now we have a Web page with, interestingly, a little bit of hidden text that you'll only see if you highlight it, which is kind of stupid. So that's not a very good Web page.

Pick your favorite color. Let's make it a little sexier.

STUDENT: Rouge.

DAVID MALAN: Okay, so, funny thing about XHTML. It only understands a fixed number of colors in words. So, "rouge," how do we get that? Well, let's compromise for a moment and just say "red," and get away with that maybe: <body bgcolor = "red"> .

(01:10:00)

All right, well, let's go back to the file. Upload this. Let me clean some of this up and get rid of this. Reload my Web page. Okay, now we have an ugly red Web page.

How can we make more precise color specifications? Well, you all had the opportunity for a recent problem set to play with Photoshop. Well, buying a $500 program to figure out what you're about to see is complete overkill, since you can look this up on any Web reference.

But if you'd like to link your recent savvy with multimedia, and now Website development, one of the things you were able to do with Photoshop, recall, was play around with colors. And you could play around with the color wheel, and the slider, and so forth. And one of the details that you probably overlooked at the time was the following. If you choose that color palette—I'm using lots of words here so we can get through this startup process—if you choose the color palette on Photoshop, which was just this background/foreground thing here

Note, on screen: clicks on Set background/foreground color on Photoshop toolbar

you get the little color wheel, once your computer's done loading. And now we have a range of millions of colors.

Note, on screen: Photoshop Color Picker

So let's pick something, oh, where will you be happy for rouge, like… ? I don't, I don't…

STUDENT: Down left.

DAVID MALAN: Down left?

STUDENT: Yeah, that's good.

DAVID MALAN: Like there? All right. So, rouge. Now, how do I get that color? I can't just write "rouge" as the bgcolor. But if you ever notice this thing down here, the pound: # sign? Well, this is an example of a hexadecimal color code.

Turns out you can write most any color, certainly that you'd care to write, in a sequence of codes, started with a sharp sign: #, followed by number, number; number, number; number, number, where each of these values is any number from zero to nine, or *a* through *f*.

Note, on board: #_ _ _ _ _ _

We talked very briefly, way early on, about hexadecimal, I think: "hex" meaning "sixteen." So "hexa" meaning sixteen. So zero through nine plus *a* through *f* gives you the ability to count from 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c, d, e, f, which is sixteen. But we're only doing it using these characters. Each pair represents a color.

Have you ever heard the acronym "RGB"? Yes? So it's sort of from yesteryear. In school, you grew up with those crazy huge projectors that projected the earliest of big-screen TVs, where you had a big red light, and a big green light, and a big blue light: RGB. Using those three colors can you create a whole range of colors by mixing them effectively, like you might paint.

So similarly can you specify colors in XHTML and in other contexts, like graphic design, using hexadecimal notation, where these two digits (first pair of digits) tell you how much red you want. These two digits (second pair) tell you how much green you want. These two digits (third pair) tell you how much blue you want.

So if I just wanted red, I could choose the largest two values here (first pair), the smallest values for these other guys, which essentially means, "Give me a lot of red, no green, and no blue."

And so in my Web page, it turns out, I could go back here and not write, "red," which is a common enough color that XHTML knows that it means. But I could say #ff0000.

Note, on screen: <body bgcolor="#ff0000">

I can save it. I can upload the file again via index.html. I can refresh my Web page, and what should I see?

STUDENT: (inaudible response)

DAVID MALAN: The same ugly Web page because "#ff0000" means the same things as "red."

Well, now let's go to rouge. So that was the request. I'll go to Photoshop, and, again, there are better ways to find this. Usually a lot of HTML books will have references in the back that tell you the codes you can type for various colors, and that's a neat trick.

I'm going to copy this (the hex code for the rouge color from the Color Picker). And I'm going to go paste it into the Web page. Again it's got to be prefixed with the sharp (#) sign. aa5961 is the value we chose.

Note, onscreen: <body bgcolor="#aa5961">

I'm going to reupload that. And I'm going to refresh. And now we have a slightly less annoying Web page to look at.

So there you have it, and you can choose from any number of thousands of colors now, using that idea. Mixing the colors manually, or frankly Google around on the Web, or use one of the recommended resources on the course Website's Resources page—though, not tonight—and you

will be able to just look these kinds of things up online. But for the most part, you can choose the words off the top of your head.

So that's an attribute: Controls the behavior of a tag in a certain way. There's only certain attributes defined for certain tags. How do you know what tags have what attributes associated with them? Memory, really, or a reference book.

So one of the things that you should realize about making Web pages is that it's completely reasonable, and expected, certainly for the course, to sit with an XHTML book by yourself, and just look things up. When you want to figure out a new trick, look it up. Not a big deal. And you do it enough times, you'll just remember it.

Alternatively, there's no need to buy books. To learn XHTML, pull up any one of the Websites that we recommend on the course's Website, or just Google around for "XHTML tutorial," or "XHTML reference." And there's so much free information, there's no need to have a book, but sometimes it's more convenient.

So we do recommend in Sets 1 and 2 an XHTML book that is a pretty good one. It's a useful quick and dirty reference to have next to you.

Well, let's do something more interesting than a Web page, like this. Let's make it white again, just for clarity's sake. Let's then... let us then restore power to the laptop. Let's change this back to white. And let's do something interesting with the body.

Note, on screen:

```
<html>
        <head>
          <title> Malan Rouge!</title>
        </head>
        <body bgcolor="white">

        </body>
</html>
```

So make a request of me—something reasonable that I can implement in XHTML.

STUDENT: (inaudible response)

DAVID MALAN: Let's do words for now. Well, no, I don't want to just type words, because I can do that pretty well. So tell me how to design something aesthetically, simple.

STUDENT: (inaudible response)

DAVID MALAN: Import a penguin?

STUDENT: (inaudible response)

DAVID MALAN: So, like, include a picture of a penguin on my Web page.

STUDENT: (inaudible response)

DAVID MALAN: Okay, you're setting the bar a little too high for me, I think. So I will use Google images. This is a good exercise. Search for Google images of a penguin. Okay, we'll go with the Linux penguin, so that there's at least some relevance here, arguably. I'm going to choose that image. Come on.

Okay, so now I have a penguin. Specifically I have the URL of the penguin. So for discussion's sake, I'm simply going to copy that URL. I'm going to go back to my Web page, and I'm going to say, "All right, I want to have in this Web page, one, I want to center everything."

And to center something, as you might be pleased to realize: , but remember to close it: .

Note, on screen:

```
<html>
        <head>
          <title> Malan Rouge!</title>
        </head>
        <body bgcolor="white">
          <center>

          </center>
        </body>
</html>
```

And the good habit to get into, honestly, is whenever you write an open tag, write the close tag, and then move back into the middle of things, so that you can continue writing.

To include an image, as you might have noticed earlier, you simply say, img, src for source, equals, and then the URL of the image: <img src = http://…. >.

I'm going save it. I'm going to go back to the FTP program, upload that. Go back to my home page, which is now far from its original design, and there's my Web page. And you can know it's mine because, one, it says "malanrouge.com" up top; but, two, it has a penguin now, and not the Malan Rouge! Photoshopped logo.

So, that's all fine and good, but what if this guy whose hosting this Website, at unix-ag.org, decides he doesn't like my using his bandwidth to make images on my Web page. This is common practice to try to include content from someone else's server in your Web page, and it's that easy.

But realistically, let's suppose for argument's sake, that this gentleman is perfectly fine with me downloading this image, right—he hasn't claimed it as intellectual property; it's just something he found on the Internet, too, so it's not copyrighted in any sense.

Well, you probably know how you can just right-click, or control-click an image in Internet Explorer or Safari. And then just do "Save Picture As." And I'm going to save it to my desktop as "penguin". And notice, what file format is it in?

STUDENT: (inaudible response)

DAVID MALAN: PNG—so that's okay. It's a PNG; might be a JPEG, might be a GIF. It's a PNG, so I don't want to change the file extension. You can't just change something and expect it to convert. I'm just going to leave it alone.

So now notice on my desktop I have penguin.png. So if I want to include this in my Web page in a more legitimate sense—that is now *I'm* hosting it and I'm not subject to what that other guy decides to do with his Web server—all I have to do is upload that file to my Website.

Now I'm going to go back to my XHTML, and say, "You know what? This isn't at a URL. It's at a local reference, which is just penguin.png: <img src = "penguin.png"> .

Note, on screen:

```
<html>
        <head>
          <title> Malan Rouge!</title>
        </head>
        <body bgcolor="white">
          <center>
             <img src="penguin.png" />
          </center>
        </body>
</html>
```

Save it. Reupload it. And what we've got next is the same exact... Whoops, that's his Website. We haven't changed his Website. I refreshed my Web page. Same thing, but now the image is coming from my server.

And this is worth bearing in mind, because sometimes people will try copying other people's images' URLs, especially from Google images, including it in their own Web page. But you can configure your Web server in such a way that, if someone tries to do that, that they just get a warning message or a broken link. Turns out this guy just hasn't configured his Web server in that way.

So the best way to do this is to save the images, copy them to your hard drive, then upload them to your own account, assuming that's legitimate for you to do so with that image.

Not bad. I made a Web page with a penguin!

Give me something else, with text this time. Yeah?

STUDENT: (inaudible response)

DAVID MALAN: Actually gave you different images.

STUDENT: (inaudible response)

DAVID MALAN: So that's quite possible.

STUDENT: (inaudible response)

DAVID MALAN: The short answer is it depends. You can write a program to do that. You can configure your Web server to do that. If you want to chat with me after, I can offer some technical details. It's nontrivial.

(01:20:03)

But it does remind me, incidentally, when we were talking about our security stuff earlier, and accessing, for instance, someone's access point that wasn't your own, or didn't belong to you.

There was this brilliant article, a few months ago, on a news Website, which explained how this guy, with way too much free time, had essentially set up a server in his apartment, or office, or wherever, that was acting as a firewall, router, switch, access point, all that stuff. But he configured this access point in such a way that anyone who connected to it, could access the Internet. And they'd get an IP address via DHCP, and everything worked just as you would expect. People could leach his Internet access all they wanted.

But he configured the access point to flip upside-down any GIF or JPEG that went through his access point, which meant anyone who accessed the Internet through his computer had this crazy, weird problem where every Web page's image was upside-down.

The other trick he played for a while was to use the same software he was running. And rather than flip the images around, he ran the equivalent of a Photoshop filter, and made all of the images blurry. So that the user visiting the Web page, you know, was checking their eyesight at that point.

So I guess, buyer beware. But you can do neat tricks, certainly, if you have control over such data. But how do you create Web pages that even have images? Well, it's as simple as this, and there are more details.

Tonight is not meant so much about teaching you every nuance of XHTML, because this would very quickly degrade into an incredibly boring talk, if it hasn't already. Rather, it's meant to introduce some of the concepts and the basic ideas. And frankly, there is absolutely no reason for anyone to ever take a course on learning HTML or XHTML. Frankly, it's the sort of thing where, after a few basic building blocks, you should be able to bootstrap yourself to learning the rest largely on your

own. Because all you have to do is turn to the page on "How do I make something centered?" and write the tag that will allow you to do that, or borrow it from something you've seen already.

What about something with text? We might want to make text big. Well, a way to make text big would be to use something called a "heading" tag. So <h1> in XHTML defines a really big heading, which just means big and bold text.

If I now say, "Welcome!" here

Note, on screen:

```
<html>
        <head>
          <title> Malan Rouge!</title>
        </head>
        <body bgcolor="white">
          <h1>
          Welcome!
          </h1>
        </body>
</html>
```

notice that, if I upload this file to the server, it will now say "Welcome!", yes, but it will say it like that, in much bigger print than you saw before.

By contrast, if I below that say not just <h1>, but go to the other extreme and say <h6>, which is another predefined tag, which you would know existed if you just looked in any XHTML reference book, or a Website

Note, on screen:

```
<html>
        <head>
          <title> Malan Rouge!</title>
        </head>
        <body bgcolor="white">
          <h6>
           Welcome!
          </h6>
        </body>
</html>
```

well, I would simply see on the line below, this.

Now you see the distinction.

Note, on screen: Welcome! (in small font size)

Well, what else can you do? Just to give you a sense of things, not an exhaustive reference.

Let me go ahead and delete that.

Note, on screen:

```
<html>
        <head>
          <title> Malan Rouge!</title>
        </head>
        <body bgcolor="white">

        </body>
</html>
```

And you'll see in tonight's lecture slides that, even though we're working through this on the fly, you have references in the form of those slides.

Suppose I wanted to change the font of something? Well, I could say the tag, and then I could say "face," where "face" equals the font name or family: <font face="

What's a font that's very popular?

STUDENT: (inaudible response)

DAVID MALAN: Arial. I'll go with Arial, because I know that one will work. By default, what font were we seeing? Do you recognize it?

STUDENT: (inaudible response)

DAVID MALAN: This is Times New Roman, which is fine for printed documents. On Web pages doesn't look so pretty. So let's say Arial: <font face="Arial

And now I'm going to say <font face= "Arial" size="

And unfortunately with a font tag, you can only say size equals 1 through 7. So I'm going to choose 6: <font face= "Arial" size="6

And these are relative sizes. It's not point size, it's not pixels. It's relative sizes, so it's somewhat browser-dependent here, too. Say <font face= "Arial" size="6"

And what's your favorite color for text?

STUDENT: Rouge.

DAVID MALAN: Something simpler. "Blue" I heard. I'll go with blue:

And (types): This is the ugly text

Note, on screen:

```
<html>
        <head>
          <title> Malan Rouge!</title>
        </head>
        <body bgcolor="white">

          <font face= "Arial" size="6" color="blue">This is the ugly text</font>

        </body>
</html>
```

Now I'm going to save it. I am going to reupload the file. And now notice that we have still a very ugly Web page, but at least now it's in Arial and it's now blue.

Note, on screen: This is ugly text

All right, so where are we going with this? Well, one, it's worth noting a couple of nuances here. So, one, just because I have Arial installed on my computer doesn't mean the next guy is going to have Arial installed on his computer.

Now it's a bit of a white lie because, these days, pretty much any computer—Mac or PC—has Arial, and it has Times, so you're safe.

But it you really like that Comic Sans font that looks like handwriting on your PC. Well, that's fine, you can call it, for instance, Comic Sans, and then your Web page will look great, or at least really cutesy on your Windows PC. Mac user pulls is up, if Comic Sans isn't installed, which it probably isn't, he's going to get some other font, probably the default font, which again speaks to this problem of a nonuniform experience.

So for the most part, what Websites will do is either, one, choose a very common font, and just leave it at that; two, they'll provide a list a list of fonts, where you can provide alternatives in this form: separate them with commas, such that the browser, if it doesn't have one installed on its machine, it will try the next, and so forth.

Note, on screen:

This is the ugly text

Or better yet, arguably, is sometimes to just describe the type of font that you want, either a serif font—and that's a font that has little twiddles on the ends. It's nice and pretty, and elegant, like Times New Roman—or something that's sans serif, like Arial, which is just straight or curved lines with no little frills on the edges of the letters.

Note, on screen:
This is the ugly text

So if I just say "sans serif," I leave it to the browser, granted, to decide what font to use, but at least I know the style of the font.

And if you look closely, for the most part, most Websites use only a handful of fonts these days. Something like Arial is common. Another one that's common is Verdana on a Windows PC, and there's a couple of others. But for the most part, it's good practice to stick with something more generic, because then at least you know the Website will look the same on multiple people's browsers.

So, looking at the slides for just a moment, just so that you have something to refer back to, this first example here is just a terribly simple example. Pretty much where we began, but it just puts in writing the most basic, perhaps, of Web pages.

Note, on screen: slide #4

This one here adds the notion of attributes and values, so you can modify the behavior of your Website's appearance.

Note, on screen: slide #5

This one here starts to get a little more involved.

Note, on screen: slide #6

So notice, we've used the font tag, which we just used on the fly, but here's a neat new feature, right? HyperText Markup Language, at the end of the day, is all about HyperText, which means links, URLs that you can click on, and so forth.

So how do you do that? Well, suppose I want to have a Website linked in my Web page? Well, I could do this.
This is the ugly text

And I will say

(types):
This is the ugly text **that leads to**

And now I'm going to say

(types):
This is the ugly text that leads to**<a href=**

for anchor.

A hyperreference here.

And my favorite Web page will be <a href="http://

… and you do need to include that in hyperlinks in the Web page, even though you don't have to type it in the browser. Close quote. Close bracket.

> Note, on screen:
>

And

> Note, on screen:
> This is the ugly text that leads to**The Weather Channel**

So I showed you this briefly, I think in the last lecture, where you have this distinction between what the hyperreference is and what the name of the link is. So notice what's in between the quotes is the URL the user's going to go to.

What's in between the open tag and the closed tag—the anchor tag—is what is going to appear on the Web page.

So if I save this, and now upload this index.html, refresh my page. Now notice I have a hyperlink in that same string of text.

And if I hover over it, notice that the URL is weatherchannel.com, but the text is something completely different.

So again are those spammers and those phishers trying to get you to click on bogus links? Well, let's just supposed that I change this to

> Note, on screen:
> This is the ugly text that leads toThe Weather Channel

But you know, I'm going to call it PayPal.com.

> Note, on screen:
> This is the ugly text that leads to**PayPal.com**

Well, what's the effect? Well, if we update the Web page and refresh it in the browser. Well, now it looks like I have a nice little link to PayPay.com. But not quite, right?

So that's how phishing attacks tend to work: When you click on one URL, thinking it's one thing, but it actually leads elsewhere.

Well, you don't necessarily just have to make text lead somewhere. You can make images lead somewhere.

For instance, the course's Website, which I would be showing you, has banners up top. And, in fact, we had some wonderful submissions for the Multimedia Problem Set, which I wanted to show you., because if you click on the banner on the course's Website, it'll flip through each of your submissions.

I'll try this just for kicks, to see if we even have a chance of seeing it.

Oh, it came up this time. So, if at home—I won't dwell on this now—but if you click this banner, you'll see each of your submissions. That's…

We'll leave that one up there for now just because it's the most startling, when someone visits the Website. It's not changing it for everyone, mind you. It's changing it just for you, using cookies, remembering what you've seen before.

What I'll invite you to do at home or at your leisure this week, is do take a look also at the several mousepad designs that were submitted. In next week's lecture, what we will do is have a vote as to which mousepad design you find to be your favorite. And then we will go to press with that. And your souvenir, come term's end, will be—I'll just pull up one, just because it makes it makes you smile. There's our winner, right? No bias here on our part.

What we'll do next week is vote. And at term's end, you'll walk away with a very nice mousepad in hand. Yeah?

STUDENT: (inaudible response)

DAVID MALAN: If it's not there, drop us a note, and we will check through the archives.

STUDENT: (inaudible response)

DAVID MALAN: It might just mean that I overlooked it.

So, with that said, back to the question at hand: How do you make links not only be based on text, but on images?

(01:30:01)

Well, so far, we've seen that XHTML is all about just nesting things, right? You can nest tags within tags, within tags, so long as you keep things balanced, at the end of the day, and eventually close those tags.

So if instead of saying "PayPal.com," I wanted a penguin to lead to badguyswebsite.com, I could simple put what here?

STUDENT: (inaudible response)

DAVID MALAN: Yeah.

(types):
<img src="penguin.png">

> Note, on screen:
>
> This is the ugly text that leads to
> **<img src="penguin.png">**

And that's it.

Oh, but this is an interesting point to make. Notice that, have I ever closed my image tag, now or before?

STUDENT: (inaudible response)

DAVID MALAN: No, and it's a little silly in this sense to do now this.

(types):
<img src="penguin.png"></image>

Because nothing really conceptually should go in between these two tags, right? What would it mean to embed something inside of an image? It's either there or it's not. It's not like it can start being there; you can put something in the middle, and then have the rest of it appear elsewhere.

So some tags like this, they conceptually don't really lend themselves to the notion of starting and stopping. But rather they're more atomic ideas: Put something here.

Well, you can do this to close it, yes, and you must… Sorry, that's completely wrong: img.

(types):
<img src="penguin.png"></img>

You must close it to be, as we've said, consistent with XHTML. Every tag that's open must be closed, but there's a trick, a shortcut. Any time you have a link that doesn't need to have anything inside of it, you can simply put your forward slash at the very end of the same tag.

(types):
<img src="penguin.png" **/**>

And we'll see an example of this again in just a moment.

So if I now save this, upload this to the Website, refresh. Now I have
This is ugly text that leads to this penguin, which in turn leads to abadguyswebsite.

> Note, on screen:
> This is ugly text that leads to (penguin image)

Notice the blue border around it (the penguin image)? That's sort of the mark of a bad Web designer. There are attributes you will learn when you start playing around yourself:
<img border="0"> will get rid of that ugly blue border around the image.

> Note, on screen:
>
> This is the ugly text that leads to
>
> **

We'll go ahead and upload this. Refresh, and now the border's gone. But it's still a link that I can click on, and be whisked away to a site that you could have bought. Doesn't yet exist.

What else can we do here? Well, notice that I've been trying to keep things nice and neat. I've been tabbing and spacing so that everything sort of flows inward and then back outward. Completely unnecessary.

You could write a Web page that is just as accurate and functional, even if you got rid of all of that (removes indents, line breaks, etc.) and left it as this.

Let me save this, reupload it. index.html. Refresh. Web page is the same.

Well, wait a minute. Suppose that I don't want the penguin *there*. I want rather below the text, because this already looks, you know, pretty ugly, right? Tonight is ultimately about how to make really ugly Web pages, right? We leave it to you to make better-looking Web pages using the same tools that we're providing you with.

So what can we do to fix this? All right, well, let's go back to the original condition, just because that's confusing.

Let's say border=0. That's where we left off.

Well, I am simply going to do this. All right, here's the start of the image. I want to put it below the text. So let's just do that, all right, just like you would in Microsoft Word. (inserts a bunch of spaces after "that leads to") Put the image down below.

All right, let's upload it again. And reload.

Uh-oh. (Web page looks the same.) Well, why is that?

STUDENT: Didn't go down enough spaces?

DAVID MALAN: All right, didn't go down enough spaces. (adds more spaces)

STUDENTS: (inaudible responses)

DAVID MALAN: They don't like that idea, Eric. Okay, what else?

STUDENT: (inaudible response)

DAVID MALAN: Good. You have to tell it. You have to break somehow. So everything we've done thus far is sort of like an implicit instruction to the browser: "Put the body here. Put the title here. Make this bold. Make this blue."

We never really said, "Put a line break here." That is to say, XHTML ignores whitespace. Any whitespace you have beyond one individual space is ignored. Which means if you want to put whitespace, carriage returns, new lines, and so forth, you have to explicitly say as much, by saying something like, "Put a line break here."

> Note, on screen:
> This is ugly text that leads to
> **
**
>
>)

or equivalently

> Note, on screen:
> This is ugly text that leads to
> **

**
>
>)

Those tags are equivalent, but it's sort of silly to write it this way **(
</br>**), so I would get into the habit of writing these atomic elements just as one entity, like that

> Note, on screen:
> This is ugly text that leads to
> **
**
>
>)

with the forward slash right at the tail end.

And now I'm going to clean this up. Because if you now know that whitespace is irrelevant, the only reason I've had this whitespace thus far is just for human readability. It quickly becomes a nightmare to read and update your Web pages, if you, yourself, can't read the code.

So we get into the habit, and we certainly preach the habit, of so-called "pretty printing"; pretty printing simply meaning "keep things human-readable," but not in a way that helps the computer. It helps you, the developer.

So I'm just going to move this over. I'm going to keep things nicely indented. I'm going to now move the link in here. This has no material effect, but it does become a little more manageable for me to maintain.

Save this. Upload it back to the server. Reload. And now we have it (the penguin image) below. Again, still ugly, but at least we're getting there.

How do I get multiple line breaks? Well, what might you think?

STUDENT: (inaudible response)

DAVID MALAN: Now let's put a bunch of them? All right, that's going to keep putting it (penguin image) down: break, break, break, break, break, presumably?

> Note, on screen:
> This is ugly text that leads to
> **

**
>
>)

And in fact, it does. And so that's the idea. Again, the concepts are important here, not the design qualities that we're preaching thus far.

Well, this is getting annoying, having to upload the file all the time, every time I make the slightest of changes.

Turns out, you don't need to do that. The beauty of having a browser on your computer is that you can visit not only URLs, but also files.

Well, it turns out, that if I put the image in the same directory as the HTML page, by dragging it there, notice that now my URL is not malanrouge.com. Notice, it's C:\ whatever the path is to the folder in which my file is.

If I refresh this, notice that now I can just edit this on the client side. And this is a neat trick, too. Because, as you develop your final projects—especially if you don't want the world seeing your work before it's completed, if only out of embarrassment—you can develop everything on your local

computer, without Internet access, assuming you don't want to download images, and so forth. Do it all in a folder. And then when you're ready, upload it en masse. Same result is achieved.

Well, what else can you do, when developing Web pages? Well, it's actually useful that the Web pages we've been developing with just XHMTL are turning out pretty ugly. Because XHTML only takes you... That's all right we don't need that. XHTML only takes you so far aesthetically.

But very commonly used today in the design of Websites is something called "Cascading Style Sheets." And this, too, is something that you'll be introduced to, largely by way of bootstrapping. We'll give you the building blocks. But really, then, refer you to, like, any online reference, or a textbook, or to this week's Sections and Workshops, where you can learn more.

But Cascading Style Sheets let you really fine-tune the aesthetics of your Web page, right? We said a moment ago, the font tag only supports 1 through 7. That's sort of lame when, what does "1 through 7" mean, if it's up to the browser.

If you want to exercise precise control, you can use things like CSS, which, just to give you a sense of it through this example, using the tag, for instance, you can create what are called "divisions" in your page. Think of these as just chunks of your page, blocks of your page that you want to do something to. Doesn't mean to draw a line or a physical division. It just means create a conceptual division.

And in this tag, notice that we have a style attribute. One way via which you can use Cascading Style Sheets, or Cascading Style Sheet examples in your Website is just to say <style=" ">.

And then anything you put in *that* value can be Cascading Styles Sheets, and that further refines the aesthetics of your Web page. And this, too... This already should look a little weird, because now it looks like the value of an attribute is, yes, something in quotes. But notice, there's a whole bunch of things now in quotes, separated by what character?

STUDENT: (inaudible response)

DAVID MALAN: Semicolon. So, for better or for worse, this is the way the world works with Cascading Style Sheets and Website development. If you want to exercise multiple degrees of control within one tag, with Cascading Style Sheets, you just separate those directives, using semicolons.

And we can sort of learn by example here.

Note, see slide #7

If this text here, let's say. It says "Hello, World." Well, this text is going to be aligned in the center. You'll see already that there are multiple ways to do things when designing Websites. You can just say text .

Or you can do it in this way, with Cascading Style Sheets.

The color of the text is obviously going to be blue. The font-family is going to be sans-serif. And the font-size—and this becomes a useful trick—is 36 point (36pt). You know what 36 point is, and ergo, the browser will render it at a fixed size, not at some arbitrary <font-size="7">, for instance. So that's useful, too.

You can do other tricks. Much of what's on the course's Website is the result of Cascading Style Sheets. So typically, when you visit... Every time I smile when I see this one. When you visit a Web page usually, a very basic Web page for years had hyperlinks that were always what color?

STUDENT: Blue.

DAVID MALAN: Blue. And if you clicked on them once, they become purple. And links were usually underlined with a blue line. Well, that's all fine and good, but it doesn't really lend itself to the sexiest of Websites, or the prettiest of Websites.

So among the ways you can control the behavior of links is with Cascading Style Sheets. For instance, every link on the course's Website is designed certainly not to be that blue Times New Roman font with a blue line under it. Rather we're using something that's sans-serif; that is red; but with no underline.

But what happens when you hover over one of the links in the course's Website?

STUDENT: (inaudible response)

DAVID MALAN: It underlines *then* to sort of draw your attention to the fact that it's a link. If we visit something like the Not Dumb Questions site, this too has links. But if you hover over these, they similarly become blue and they become underlined there.

(01:40:07)

Cascading Style Sheets are a wonderful thing, and this is just a snippet, a sneak preview of what you can do with Cascading Style Sheets. We'll defer more of it to the Section that's devoted to enhancing your Websites with Cascading Style Sheets, or to any number of online resources, a couple of which we link to in the Resources page. But for now just think of it as a way of exercising more aesthetic control over your Website.

For the most part, the Websites you design for the course, and certainly the stuff that we've looked at tonight, are largely static in nature. That is to say, you design the Web page, you make it look like something, and I'll wave my hands at this as another example of CSS. But it's too much, I think, to throw everything at you in the first night, before you've even written the XHTML.

Well, most of the stuff we've looked at is in the form of static content. You make a Web page and it looks the same every time you pull it up.

Well, many Websites these days, certainly the ones we visit the most commonly, like Amazon and Google, they're ever-changing. And certainly, for all of the billions of Web pages that Google has indexed, there is not some guy writing an XHMTL Web page that, you know, gives you a link to the first ten results, and another guy who wrote the page for the results 11 through 20, and another guy who wrote the results for, you know, 21 through 30. All of that stuff, needless to say, is automatically generated.

And just so that you've heard some of the jargon, among the technologies that these bigger Websites are using—none of them are hard, and they just require a bit more savvy than you can introduce, certainly, in just one lecture's time.

Server Side Includes look like this—again, I'll wave my hand at it for now, but this is something you can return to with your Final Projects, if you're so inclined—allow you to output things dynamically, like the current time, or the user's IP address, or other such things that the server would know.

Note, see slide #11

DHTHML.

Note, see slide #12

This is the combined use of something called JavaScript—a programming language—with XHTML, and CSS, Cascading Style Sheets, that creates those Websites that have more interactivity: things you can drag and drop. And any Website that's a little sexier than the more static ones you've seen often uses the combination of these technologies.

Something called AJAX is related.

Note, see slide #13

And again, this is something that might recur if you continue in your progression of studying of programming, and so forth.

Note, see slide #14

CGI is the name given to a technology that allows you to generate Web pages dynamically, often on Unix or Linux computers. The course's Website uses CGI to dynamically generate the Lectures page and the Sections page. We rarely edit XHTML ourselves on the Website. We edit a configuration file, type some commands, and bam, the Website changes itself to save us the time of having to write XHTHML manually.

On Microsoft platforms you have things like ASPs.

Note, see slide #15

On Java platforms, you have JSPs, and there are other such dynamic technologies.

Note, see slide #16

We will return to more of what you can do with your Final Project as the course progresses.

Let us make one quick announcement about this week's Workshop and Section, if Eugenia wouldn't mind coming down for just a second.

EUGENIA KIM: Hello. So tonight's and Saturday's Sections are on XHTML. So basically you'll be working on actually making a Web page. And then this Saturday's Workshop is an introduction to digital video editing. So any of you moviemakers out there, we look forward to seeing you.

DAVID MALAN: Excellent. So tonight again, to be clear, was meant to introduce some of the concepts and some of the basic building blocks of Website development, and where you're going with your project. It's through these Sections and Workshops, and also on your own. Over the next two months will you get an opportunity to really dive in deep; look something up online; ask questions of us; figure out, how do I do this?

But one of the best ways to learn, to be honest, about how to write Web pages is to just look at the source code of other people's Web pages. If you see something you like—not to copy it, per se, but to look at how they implemented it, and to learn from others. And that, frankly, is certainly how I learned this stuff early on. Someone gave me a one-hour crash course on HTML, and I went off and learned the rest on my own, just by asking questions and poking around.

So you have many months to work on this. We'll return to it over time. But until then, we will see you next week.

(end)

(01:44:18)