BRIAN YU: All right, welcome, everyone, to Web Programming with Python and JavaScript. My name is Brian Yu. And in this course, we'll dive into the design and implementation of web applications.

In lectures, we'll have an opportunity to discuss and explore many of the ideas and tools and languages that are central to modern web programming. And through hands-on projects, you'll have an opportunity to take those ideas and put them into practice, designing multiple web applications of your own, culminating in a final project of your own choosing. Throughout the term, we'll cover a number of topics in this world of web programming, beginning with HTML5 and CSS3, two of the languages that are central to the understanding of web pages. HTML is a language we'll use to describe the structure of a web page. And CSS is the language we'll use to describe the style of a web page, the colors and the fonts and the layouts and the spacing that make the web page look exactly the way we want it to look.

After that, we'll turn our attention to Git, a tool not specific to web programming per se, but that we can use in order to version control our programs, to keep track of the different changes we make to our web programs, and to allow us to be able to work on various different parts of the web application at the same time before merging those pieces back together. After that, we'll take a look at Python, one of the first main languages that we're going to be exploring in the course, which is the language that we are going to use in order to build our web applications. Specifically, we'll use Python using a framework called Django.

Django is a web programming framework written in the Python programming language that we're going to use to make it easy to design and develop our web applications Django in particular makes it easy to design web applications that interact with data. So after that, we'll turn our attention to SQL, a language that we can use to interact with databases, in particular looking at how Django allows us to use models and migrations to interact with data and allow users to interact with data all the more easily.

Next, we'll turn our attention to the second of the main programming languages that we'll be exploring in this class, JavaScript, and looking at how we can use JavaScript to run in users' web browsers to make web pages just a little bit more interactive. In particular, we'll use JavaScript in the context of user interfaces, looking at modern user interfaces and exploring how it is that those interfaces work and how we can develop those user interfaces with a combination of Python and JavaScript.

Next, we'll turn our attention to testing and CI/CD, or Continuous Integration and Continuous Delivery, which are tools that we can use and software best practices to make sure that we're able to design and develop code more efficiently. And testing in particular makes sure that as we make changes to our code, we're not breaking existing parts of our web application by making sure that we have a whole suite of tests that we can use to ensure that our web application is always behaving as it should.

And finally, we'll turn our attention to scalability and security on the internet, thinking about what happens as our web application grows larger. As more and more different users start to use our web application, how do we load balance between those people? And what do we need to change about our database to make sure lots of users are able to connect to our web application at the same time? Moreover, we'll look at the security implications behind designing our web applications. What might an adversary do if we're not careful? And how should we proactively be designing our web application to make sure that it's secure?

But today, we begin our conversation with HTML and CSS, two of the languages that are foundational to understanding web pages and how web browsers are able to display those web pages. And we'll start with each HTML, or HyperText Markup Language, which is a language that we can use to describe the structure of the web page, all of the buttons and the text and the forms and other parts of the web page that the user ultimately sees and interacts with.

Our very first HTML page is going to look a little something like this. It's going to be text-based code that we write that a web browser, like Safari or Chrome or Firefox, is then able to look at, parse, understand, and display to the user. So let's take a look at this page one line at a time and get an understanding for how it works.

Even if you don't quite understand all the nuances of the syntax, there are probably a couple of things that stand out to you. You might notice the word title, which probably reflects the title of the web page, for example, which in this case appears to be the word hello. And then down further below, we see that we have the body of the web page that seems to contain the words hello world.

So what is this web page actually going to look like? Well, let's take a look at it. We'll go ahead and open up a text editor. You can use any text editor you want. But for this course, I'm going to use Microsoft's Visual Studio Code. And I'm going to open up a new file that I'm just going to call hello.html.

Inside of hello.html I'm going to write the same HTML that we just saw a moment ago. And we'll explain each of these lines in due time. But recall that we had a title of the page that said something like hello and then a body of the page where we said something like hello world, for example.

So this is our very first HTML page. And if I go ahead and open that HTML page as my opening hello.html HTML, for example, inside of a web browser, what I'll see is something like this. In the body of the page, I see the words hello world. And if you notice up here at the top of my web browser, I see the title bar, where I have the title for this page, which in this case is just the word hello.

So this is our very first web program that we've been able to develop just using HTML. And now let's explore in more detail how exactly this program works. So here again was the web page that we were just looking at. And this very first line here, DOCTYPE html is what we might call a DOCTYPE declaration. It's a way of telling the web browser what version of HTML we're using in this particular web page, because depending on the version of HTML, the web browser might want to display different information or it might need to parse the page a little bit differently. Each version of HTML has had a slightly different way of indicating that version. But this line here, DOCTYPE html is our way of saying that this HTML page is written using HTML5, the latest version of HTML.

After that our HTML page is structured as a series of nested HTML elements, where an HTML element describes something on the page. And we might have elements that are inside of other elements. Each of those elements is indicated by what we're going to call an HTML tag, enclosed using those angled brackets. And right here, we'll see the beginning of the HTML tag, which means that this is the beginning of the HTML content of our page.

Down below this slash HTML means that this is the end of the HTML content of the page. And in between is the actual HTML content of the page, which might include other HTML elements. You might also notice that in this HTML tag we've specified what we're going to call an HTML attribute, some additional information that we're giving about this tag. In particular, we're giving it a lang, or language, attribute, which is equal to en, or English. This just tells the web browser or anyone looking at the HTML of this page that this page is written in a language, and the language it's written in is English. And this is helpful for search engines, for example. When they're looking through many different web pages trying to figure out what language each web page is in we can just tell the search engine or anyone else who's looking at the page that this page is written in English.

Now, inside of the HTML body of the page, we have a number of different elements that are going to describe what we want on this page, starting with the head section of the web page, which describes stuff not in the main body of the web page, the part of the web page the user sees, but other information about the web page that's going to be helpful or useful for web browsers to know about. For example, one important thing that a web browser needs to know is, what is the title of the web page?

And here, we see a title tag, again, indicated by the word title in those angled brackets, followed by the end of the title tag, indicated by a slash before the title. And in between the two title tags is the word hello, which means the title of this page should be the word hello. And that's all the information we'll have in the head of the page. We'll add more information there later, but for now all the web page needs to know is that it has a title and the title is the word hello.

Next up comes the body of the page, again, indicated by a body tag and that ends with a tag with slash body, meaning this is the end of the body of the page. And the body of the page, again, is just the visible part of the page that the user can see. And what do we want inside the body of the page? For now, we just want the text, hello world. And that's the information that's going to be displayed when someone visits this web page.

And so that's all there really is to this HTML page. We specified in the header that there's a title of the page called hello. And inside the body, we're saying the page should say the words hello world.

And if you want to visually think about the way that all these HTML elements are structured, it can sometimes be helpful to think about an HTML page in terms of a tree-like structure that we call the document object model, or DOM. And so, here, for example, is what the DOM for this web page might actually look like. Here on the left is the HTML content that we just saw a moment ago. And over here on the right is the DOM, the document object model, the tree-like structure that describes how all of these HTML elements are related to each other.

So we start up here with the HTML element. And this parent element, so to speak, has two child elements within it, a head element and a body element. As we can see here, we're inside of HTML. We have a head section and a body section.

And the indentation here that we're including in the HTML text, it's not strictly necessary. The web browser doesn't care what the indentation is. But it can be helpful for someone who's reading the page just to see the indentation to understand visually that the head is inside of the HTML element and the body is inside of the HTML element too.

So inside of the head element, we have a title element. And inside of the title element is just the text, the word hello. And likewise, inside of the body element, we also have some text, the text hello world. So thinking about HTML and HTML documents in terms of this structure can be helpful for understanding which HTML elements are inside of which other HTML elements. And that's going to make it easier for us to reason about these pages later on. And especially as we later transition into the world of JavaScript, JavaScript is going to make it all the more powerful and give us the ability to actually modify parts of this DOM as well. But we'll certainly get to that in due time.

So now, let's take a look at some of the other common HTML tags and HTML elements that we might be interacting with in our web page. And we'll start by thinking about HTML headings, so big banners at the top of the page, for example, some headline that describes what a page is about. So I'll go ahead into my text editor and create a new file that I'll call headings.html.

And the structure of this page is going to be pretty similar to the pages that we've seen before already. So I'm going to start by just using the hello.html text and paste it in here. I'll change the title of the page. Instead of hello, we'll go ahead and call it headings. But inside the body of this page now, I want something a little bit different. I'm going to, inside the body the page, use an h1 element and say this is a heading, for example.

So h1 is a tag that I can use to create a large heading at the top of my page, like for the title of the page, for example. So if I open up headings.html. I might see something that looks like this, a big heading at the top of my page that says, this is a heading.

h1, where the h stands for heading and the 1 stands for the largest possible heading. And in fact, HTML gives us a number of different tags that we can use in order to create headings of various sizes. So, for example, I could also say h2 inside of which I say, this is a smaller heading. If h1 is the largest heading, h2 is the second largest heading.

So if I load this page, for example, I now see the h1 at the very top. This is the big heading. And then beneath that, I see this is a smaller heading, the h2. And it turns out there's also h3, h4, h5, all the way down to h6, which is the smallest heading, such that if I load this page now, I have a big heading, a smaller one, and then here's the smallest.

So we can often use these h1, h2, h3 tags just for visually organizing text inside of a page. If I want the title of the page, but also I want titles for each of the various different sections and subsections that might be contained within that page as well. So those are headings.

And now, let's also take a look at some other elements that we might want to add. On web pages, we see not just titles and not just text, but we might also see lists, for example. Like if you've ever used a to do list program on a web page, for example, you might see a list of things that you need to do or other web pages might display lists of information.

And it turns out that HTML has two basic types of lists. We have ordered lists for things that are in a particular order, like item number 1, item number 2, item number 3. And we have unordered lists for lists that don't have any particular order. So just bullet point, bullet point, bullet point, for example. And both are quite easy to use.

I'll go ahead and create a new file. And we'll call this lists.html. And again, in list.html, I'll copy the same structure from hello.html. We're again going to have DOCTYPE html just to indicate the version of HTML. Most of the heading is the same. I'm just going to change the title from Hello to Lists. And then we're going to replace the body of this page to show some different information here.

So let me first show what an ordered list might look like, something that has numbers, 1, 2, 3. An ordered as an HTML tag is just ol, ol for ordered list. So I can add a tag that says ol.

And now inside of my ol element, my ordered list element, I need a new element for every list item. List item we're going to abbreviate to just li. So the li tag in HTML is what we're going to use to designate an item inside of an HTML list.

So here, for example, I could say li and then first item. Then I could do the same thing, li second item and then again li third item. So what I have here are some elements and then elements nested within other elements. I have an ordered list element inside of which are three other HTML elements, three list items that are each indicating each of the individual items that are inside of my HTML list.

I can now open this up by opening lists.html. And this is what I see. I see an ordered list, where I have item number 1, first item, second item, third item. Note that I didn't actually need to in the HTML anywhere specify the number 1, the number 2, and the number 3. When my web browser reads this should be an order list, my web browser, Chrome in this case, just adds those numbers in for me because it knows what an ordered list means and it knows how to take the HTML that I've written and display it in the way that I intend to the user.

Now, in addition to ordered list that all have numbers, 1, 2, 3, we also have unordered lists that are just bullet points, bullet points of information. So I could, up above, add some more content to this HTML page. I can say here is an unordered list.

And just as an ordered list we represented using the ol tag in HTML, ol standing for ordered list, likewise we can use the ul tag in HTML to create an unordered list, u for unordered. So here, we're going to add a ul tag. And again, my text editor here is automatically adding the closing tag here, this slash ul, meaning the end of the unordered list. And many text editors will do this now just so you, the programmer, don't forget to add that.

And now inside of this unordered list, we're again going to have some list items. Also, using the li tag, here is one item. And here is another item. And here is yet another item.

If I go ahead and refresh the page now, I'm still on list.html, I now see that on top of my ordered list, I have an unordered list, where each item instead of being numbered 1, 2, 3, is instead labeled with just bullet point, bullet point, bullet point, where each of these bullet points and each of these numbered items is a list item element, or an li. So hopefully now we can see that as we start to explore these various different HTML tags and nesting HTML tags inside of one another, we're able to create more and more interesting web pages as a result.

So let's explore now what other types of web pages we can create using other types of HTML elements. In addition to lists, one thing you might imagine is that one of the important things on the web is not just displaying text, but also displaying other types of media, like images, for example. So how might we go about doing that?

Well, I can, for example-- let's go back into my text editor. Let me create a new file that I'm going to call image.html, which is going to contain some code for displaying some images. I'll go ahead and go into hello.html and copy this text into the page, again, change the title to Image. And now, inside of the body, I'm going to add a new tag called image. And the image tag has a couple of required attributes.

Remember that attributes are what we saw before things, like adding a lang equals en to the top of my page to indicate that this web page is written in English, for example. And the image tag has a couple of required attributes that I need to add. In particular, when I display an image on the page, I need to specify what image I actually want displayed, for example. So I might specify image src, short for source, is going to be equal to what image do I actually want to display on this page. And it just so happens that inside of my folder where I have image.html, I have an image called cat.jpeg. So I'm just going to specify cat.jpeg as the file name of the image that I want to display.

And it turns out that images also have a second required attribute. In addition to the file name or the link to whatever image I want to display, I also need to provide some alternative text, a text-based representation of what you're going to see on the image because in some cases, some web browser might not be able to render the image correctly. You might imagine if there's some error rendering the image, or if someone's on a slow internet connection, or if someone's using a screen reader and therefore can't actually see the image. We want some text-based representation of that image as well.

And so I'll provide some Alt text, some alternative text, that can be used to substitute the image in case for some reason we can't display the image. And the Alt text that I'll use in this case is just going to be the word cat, for example. And that's all I need.

Now, notice in particular there's something a little bit different about this image tag compared to other tags that we've seen before in the sense that it doesn't have a closing tag in the sense that the body had a beginning of the body and an end to the body, our ordered list had a beginning of the ordered list and the end of the unordered list with list items in between. It doesn't really make sense for an image, for example, to have the beginning of the image and the end of the image and some content in between, because the image is just a single HTML element that can't really have anything inside of it.

So in that sense, we don't actually need a closing image tag. The image tag is self-closing. It is its own beginning and end. So we can just say we want an image to be here that is cat.jpeg with an alternative text of just the word cat, for example.

So now, if I open up image.html, we'll see that what gets loaded is quite large, a picture of a cat. And I can scroll around and see this entire image. Of course, this picture of a cat is probably larger than I wanted it to be. I probably, when I my user visits this web page, I don't want them to have to scroll all the way to the right in order to see the entire cat. So I can actually add additional HTML attributes in order to modify the size of the image that I'm displaying. And later we'll see we can use CSS to do a similar thing as well.

But for now, what I can add is an additional attribute and say that let me give cat.jpeg, this image tag, another attribute that, in this case, I'll just call width. And I'll say that width is going to be equal to 300, because I would like for this image to be 300 pixels wide, for example. So now, if I refresh this page, I now see that the same cat image appears, except now it appears at 300 pixels wide exactly. So I'm able to add additional attributes, additional information to control how an HTML element is going to appear. In this case, I want to control its width, and it automatically scales down the height to make sure that the image is proportional as well.

Now, on the internet, in addition to just displaying information on a single page, it's also common for a page to link to other pages. In fact, that's one of the main important values of the internet is the ability to go from one page to another via these links. And so one thing we might reasonably want to do is add some links to our page, where if you click on something, you're taken to another page altogether. So let's take a look at an example of that.

I'll create a new file based on hello.html. And I'll add lang equals English for good measure. And I'll call this new file link.html, where here, we're going to practice with building some links into our HTML page. I'll copy the content of hello.html again, call this link.

In order to create a link, I'm going to use a tag called the a tag, short for the anchor tag. And the a tag takes one important attribute, which is called href, for hyperlink reference, which is going to specify what page I would like to link to. So if, for example, I wanted when a user clicks on this link to go to google.com, then I'd set the href attribute of this tag equal to HTTPS://google.com, for example.

Then inside of the a tag, I would specify what text I want to display. What text should the user see such that when the user clicks on that text they're taken to the web page? In this case, I'm just going to say something like, Click here, for instance.

Now, if I open up a link.html, this is what the user sees. They see a blue link that says Click here. And when the user does click on that link, they're taken to HTTPS://google.com.

And it turns out that we can use this href attribute not only to link to different websites altogether, but we can link to different pages on the same website. So, for example, if I wanted to link to that cat page that I designed a moment ago, instead of linking to google.com, I could instead just link to image.html. And now, if I save that and refresh, or if I open up link.html again, now I see a click here link, where when I click on Click here, now I'm taken to that page, image.html, that happened to have that picture of a cat from before. So using these anchor tags and href attributes, we're able to link together multiple pages. So that if we have a website that has many different web pages, we can connect them all together by using a combination of these various different links.

So now that we've seen images and links and lists, what other HTML elements might we add to our web page? Well, one thing we might want to add are things like tables, just other ways of displaying information. So let's go ahead and create a table and look at what HTML elements we can use in order to do so.

So I'll go back here to my text editor, create a new file called table.html. Using the same starting HTML, we'll call this page Table. And inside the body of this page now, there are a number of different HTML elements that we'll need in order to create a table, because as you might imagine, a table is really composed of multiple parts. We have our big table. But each table is really just a sequence of individual table rows. And each of those rows is really just a sequence of individual cells of data within that table.

And so that structure that we're imagining, a table that consists of individual rows, where each row consists of individual cells, is exactly how we're going to represent this table in HTML. We're going to start with just a table tag. That's going to represent the entirety of this table.

But inside of the table, we might have different parts. We might have the heading of the table. We might have the body of the table. So in order to represent that, I'll add thead. That's going to stand for the heading of the table, the stuff at the top of the table that might indicate what each column of the table means, for example.

And let's see, what columns do I want? Well, let's go ahead and add some table headings, which I can represent using the th tag, h for heading. And maybe I want in this web page to display information about various different oceans, for example. So maybe I have one column for the ocean and another column, another table heading, for the average depth of that ocean, and another table heading for the maximum depth of that ocean. And that'll be the very first row of that table, the heading of the table.

But in addition to the heading of the table, we also have the body of the table. So underneath the thead, I'll go ahead and include tbody, body for the main part of the table where all my data is going to be. And that body is going to consist of individual rows of a table. So I might have a tr, which here stands for table row. And inside of this table row, we'll go ahead and add some individual data points inside of the table.

So inside of my table row, I'm going to have one table data point, or td, for table data, that says Pacific Ocean, for example, then another table data that says 4,280 meters, and then another one for the maximum depth of the Pacific Ocean, which is 10,911 meters. And, in fact, these three table heads as well at the top of the page, the ocean, the average depth, and the maximum depth, those should actually probably be in a row of their own as well, because the very first part of the table, that is also a row. So I'll go ahead and add a tr, short for table row and inside of that tr put these headings.

I'll go ahead and add one more row just so we can see what this looks like. And then, we'll take a look at the page and then go back to this code. I'll add the Atlantic Ocean, too, which has an average depth of 3,646 meters, as well as a maximum depth of 8,486 meters as well.

So when I open up table.html now, here's what I'm going to see. I'm going to see a table-like representation of the data. It's not just one thing after another after another anymore. It's structured in a table. Now, granted, there aren't any borders. And I could probably add some colors and spacing to make this look a little bit nicer.

But I see three columns-- ocean, average depth, and maximum depth, where this very first row is what we might call the table header, the very top of the table that's defining what all of the columns mean. Inside of that table header is a single table row that has three table data cells-- ocean, average depth, and maximum depth. Then beneath this table header, represented in bold, is the table's body, or the tbody element, inside of which we had two rows, one for representing the Pacific Ocean, one for representing the Atlantic Ocean. And then we had data cells in each one of those rows for representing each of the individual cells that's located within this table.

So this is what that page ultimately looks like. And let's take one more look at the HTML just to get an understanding for how all of these tags interact with one another. And no need to memorize all these tags right now, slowly as you begin to design HTML pages, you'll start to get more familiar with what HTML tags are available to you. And certainly, all of these HTML tags are things that are easy to reference if you need to look them up. It's very helpful to be able to look up something like, how do I create a table in HTML? And then you'll be able to see what the various different tags you'll need to add are in order to generate the table that you're looking for.

But, again, just to recap, here, we have a table element, inside of which are two child elements, thead and tbody. Inside of each of those are one or more table rows, representing using tr. Inside of each of those are three table data cells, representing using td. And so using these nested tags, elements inside of elements inside of other elements, we've been able to build something far more complex than just a bulleted list. We've been able to build an entire table that has information as well.

But ultimately, our web pages should be web pages that don't just display information, but that also let users interact with that information in some way. For example, you might imagine that on Google's home page, for example, it's not just unchanging. There's a field where I can type something in. And anytime users can provide input to a web page, we generally call that a form, or some place where a user can fill out a form in order to provide information to the web page.

And so now, let's take a look at how we can use HTML in order to create a form that's going to display some information. So I'll go ahead and create a new page called form.html, again using that same HTML as before. We'll call the page Form.

And inside of the body of this page now, let's say that I want to create a form that gives the user an opportunity to provide their full name, for example. How do I do that? Well, the first thing, I need is a form element, some way of saying, here is going to be a form.

And now, inside of that form, what are the various different parts of the form? Well, there's really two parts that you might imagine to this form. One is a place for the user to actually type in their name. And they probably also need some way to submit the form, some button that just says submit, such that they can click on that button in order to submit the form.

So how would we do that? Well, in order to create an input field, we're going to use an input tag, who's type in this case is going to be text. There are a number of different ways that users might provide input to a form. They might type in text. They might choose from a dropdown menu. They might choose from a radio button option. Or they might provide input as by clicking on a button, for example. In this case, we're specifically using the type attribute to say that when the user is providing input in this way, the type of input that they're providing is going to be some kind of text.

Then, we might provide a placeholder, some default text that's going to be inside of that input field the first time the user looks at the page. So, for example, the placeholder might be Full Name. That way the user knows that what they should type into this placeholder is their own full name.

And then finally, we're going to go ahead and give a name to this input field. Now, this isn't going to be something that the user sees when they visit the page. But anytime you submit a form, when we receive that forum in our web application-- something we'll explore later on-- we need some way of knowing which input field corresponded to which value. And so we're going to name each of the input fields just so that later on we'll be able to reference them. And, for now, since the user is typing their full name here, we could just name this full name. Or we could more succinctly just say name as the Name of this input field.

After that, we have an input field where the user can type in their name. And now, we need some way for the user to be able to submit this form. So we might say something like input type equals submit to say, here's a way for the user to submit the form, type equals submit means. This is how they're going to submit the form when they're done completing it.

Now, if I open up form.html, this is the page that we're ultimately going to see when we load this HTML. This entire page just contains a single HTML form. But that HTML form contains two parts. The first part was this input element here that allowed an opportunity for the user to type in their full name. They type in their full name into this input field. And when they're done, they can click this Submit button to indicate that they would like to now submit this form.

Of course, right now, this form isn't going to do anything when we type in our name and click Submit, because we have an added and a logic in order to handle this form. But later on, as we transition into the world of building web applications using Python, we'll see how we can design a form, such that after the user submits it, we save information to a database or display some sort of results back to the user, all by using the power of building these web applications and connecting them to these sorts of HTML forms.

And HTML forms can actually get quite a bit more complex. We'll take a look at another example. For instance, let me open up form1.html, which is a form that I built in advance, which shows a number of other ways that users can provide information as input to an HTML form. Here, we see an input whose type is text, meaning we want the user to type in their name as text.

But you might also imagine that if a user is logging into a website, for example, they might in addition to typing in a text-based name or username or email, also provide a password. And generally, if you've been on a website and you've typed in a password, the password characters don't all show up as the actual characters. For security reasons, they generally show up as just little dots on the screen hiding the actual characters that they're representing. And in HTML, we can do that very easily by just saying that the type of this input is password. If they're typing in a password, our web browser will know not to actually display those individual characters.

In addition to just text-based input, we also have radio button input, as I alluded to a moment ago. So here, we have a number of different radio inputs, where the user might be able to select from a number of options, choosing their favorite color, for example, from a number of these options.

And finally, just to take a look at one other additional feature of HTML5, in fact, a new feature of HTML5, is something we might call a data list, where we might have the user choose in a dropdown from a number of different options. But we want to very quickly filter down or autocomplete based on those options. So if the user needs to select what country they're from, for example, we might have an input field and specify that it's going to be associated with a list called countries. Then immediately below that I have a data list element whose ID is countries, where here, I'm going to specify these are all of the options for what country we could have. Each one is inside of an option element whose value is whatever country they could select. And we have all of the countries of the world listed in these option elements. So this input here is going to allow me to select one option from a list of all of these possible options.

So now, if I open up form1.html. Here's what that form ultimately looks like. I can, here, inside of the name field-- again, that word name shows up, because it's the placeholder-- I can type in my name here. And inside of the password field anything I type is going to show up as just little dots instead of the actual characters, because the type of that input field was password, instead of the type being text.

In favorite color, I now have the ability to choose between various different favorite color options. In a radio button format, I choose from one of a number of options. And finally, inside this country dropdown, I have the ability now when I click on it to see all of the countries, but as I start to type letters, like u-n-i-t, it filters down to only the options that I actually care about. So here if I type in enough letters, eventually I see United States. And I can click on that option as well.

So HTML5 builds in these additional features to make it easy to implement something like a text field, where it will auto complete based on the text that you provide. You can just specify that it is inside of this data list and then provide a list of all the possible values. And then HTML and your web browser in turn will take care of the process of rendering that information in the way that you expect it to be displayed.

So those are just some of the possible HTML elements that we can ultimately create by using these various different elements that we nest within each other. And there are definitely other HTML elements that exist as well that you can begin to explore. But all of them follow a very similar pattern that we're going to have some tag, that might require some attributes, additional information about the HTML, to give context to the web browser for how that element should be displayed. Maybe that element needs to have a particular source for an image. Maybe it needs a link in order to link to somewhere, or other information as well.

And then inside of that element, you might nest other elements. So that your table has rows. And inside of those rows, we have other cells. And you might imagine nesting elements inside of other elements inside of other elements.

But right now so far, all of our web pages have been rather simple. They've just been-- we've described the structure of the page, and we've described we want a list here, we want to form there. What we might really like is some way of specifying that we want to style our web page in some way. We want to add color. We want add spacing. We want to add some sort of other layout to our page as well.

And in order to do that, we're going to use a second language that we're going to call CSS, short for Cascading Style Sheets. In particular, we'll use the latest version of CSS, CSS3, which gives us the ability to take an HTML page and tell the web browser how we would like it to be styled. Instead of just black text in the same font on a white background, we can begin to specify particular CSS properties of how we would like this page to look to make sure that the page looks the way we want it to.

So let's take a look at a simple example now of CSS to take a look at how we can add some CSS code to our page. So I'll go ahead and create a new file that I'll call style.html, just to demonstrate some basic ideas of adding some style to our page. And we'll go ahead and copy the same hello.html from before.

And maybe in addition to hello world, I display in h1 some big heading at the top that says like, Welcome to my web page, for example. So now, if I open up, style.html, this is what I see. I see a big heading at the top that says welcome to my web page, beneath which is just the text, hello world.

And now imagine that I want to add some style to this heading at the top of the page. Maybe instead of being left aligned, I want it to be centered. Maybe instead of just being black text, I'd like to change the color.

In order to do that, just as we've used attributes in the past to add additional information to this particular HTML page, we can do a very similar thing with CSS. We can specify that we're going to give this h1 element a style attribute. And that is going to be equal to.

And then in quotation marks, we're going to provide all of the CSS properties that we would like to add to this particular element. So the way that CSS styling works is that we can give elements individual CSS properties, where a property is something like the color of the element or the alignment of the element. And each of those properties has a default value. But we can change its value to something else.

So if, for example, I wanted to change the color of this heading, so that instead of a black heading, it displayed as the blue heading, I could say for this h1, I would like to give it a color property. And then to give the color property of value, I say color colon and then the value that I would like to give to it. So color colon blue, for example, followed by a semicolon will change the color of this h1 element to blue.

And my text editor is automatically showing me a little square that shows me what this color blue is actually going to look like. This isn't part of the text. It's just my text editor being helpful so that I can see in advance as I'm writing this code what the color is actually going to look like.