標題前的數字(如 5.)表示對應的 slide 的號

0. Introduction to the Course

Welcome to HTML, CSS, and JavaScript. This course is the first in a series of courses which make up the full stack of web development specialization.

So, this quick presentation, you'll appreciate some context for the course and also you'll know what you have to develop for the three assessment tasks on the course.

So first some context, the Internet. Well, the Internet is comprised of many servers and many browsers which grab information from the servers and then display them. This course is focused on the left hand side, the browser.

So, if we regard the browsers and the servers as a series of layers, this particular course is focused on this third layer, which is shown in brown, and is labeled as client side code. That's what we're focused on in this first course.

Now, if we add extra skills from other courses, then we'll understand the right hand side, which is server. And also upper left side, jQuery, Bootstrap, angularJS, and a lot of other things as well. But those are all in other courses. This first course is focused on the bottom left corner, CSS, HTML, and JavaScript. Let's zoom in on that diagram a bit more, and we get this. So, HTML is the visual layer. Text, image, video. How we create the things that the person sees, who's looking at a browser. Above HTML, we have CSS. CSS sets the display rules and other types of rules for HTML.

On the right hand side we have JavaScript. JavaScript can change any CSS or HTML at any time. These are three technologies we focus on in this, our first course in the full Stat Web Development Specialization. So, let's have a quick look at the three assessment tasks that you have to do on the course. The first one looks like this. It is a web page for a dating website company. You have to build this using html and CSS, and there's a few interactive elements as well. For our second assessment tasks it is a guessing game. And this introduces us to functions and loops and variables, all the typical things that you have with any programming language. We use those to make a guess the color guessing game.

For our third assessment task it is also mainly based on JavaScript. And it is a little bit harder than the previous two tasks. And for this you have to build a find the missing face kind of game. This game has a series of levels. As you finish one level, it gets harder and you go to the next level. So, that's a quick overview of context for the course and also the three main assessment tasks for the course.

Full Stack Web Development Specialization

Welcome to the Full Stack Web Development specialization. The Internet and the web was hardly known to the general public as recently as two decades ago. Absolutely, and in the last five or ten years alone there have been huge developments. If we consider browser technologies, we have HTML for text display, CSS for styling, and JavaScript for intelligence and behavior management. All of these have had huge developments very recently.

Websites are increasingly becoming web applications. Added to that, people are using mobile devices to access the Internet increasingly.

All this means that we need to have multi-platform support. And, this is where server side support for targetting multiple platforms is getting very crucial.

And that's why we've developed our Full Stack Web Specialization. The ultimate goal is to develop a capstone project which includes web, mobile, and server side components all working together seamlessly. To achieve that we have a series of courses. The first course teaches you HTML, CSS, and JavaScript. And then we move on in the second course to jQuery and the Bootstrap framework. >> In the third course, we will cover web applications, using AngularJS. In the fourth course, we will look at multiplatform mobile application development using HTML, CSS, and JavaScript. Using platforms like Córdoba and Ionic Framework. In the fifth course, we will move on to the server side, to look at node.js and node.js frameworks and MongoDB.

All this will lead you to the Capstone Project, where you will make use of the skills that you learned over the five courses. To develop a web, mobile and back-end support based project.

Module outcomes

By the end of this module you will be able to

Create web pages using a variety of different HTML elements
Change the display and behaviour of those HTML elements using CSS

1. Learning About HTML

So now we have a series of videos exploring HTML.

Now, HTML is essential for any web developer. You have to have a good understanding of it before you can move on to JavaScript and other things which use HTML. So, it's a little bit dry perhaps, a little bit kind of boring even, but you have to appreciate it before you can start building using it.

So, how do we learn HTML? Well, I have a series of slides. The slides give the background knowledge of HTML. And also there's lots of examples, which we go through. All those examples are available on the website. And at the end, there's an assessment task which explores your understanding of CSS and HTML.

Now, what about if you already know some HTML, which is quite common. Lots of people pick up different pieces of HTML. Well, if I was you, I would check the start of each video. I would check the slide which says, after this presentation, you will know about and then you'll know the target of that video.

Also, in the first few slides, there's a list of HTML elements and attributes, and also style properties which are discussed in that video. So, if I was you, I'd probably check those slides. If you know all of those things written, then that's a good sign you could skip the video. But if you don't know, it's probably an indicator that you should stick with the video and go through it.

Here's an example of the kind of thing that you'll see. This is just a list of elements, and this is a list of style properties. So, this is the kind of list you'll see near the start of each presentation.

2. About HTML

All right. So let's look at HTML in a kind of overview way first before we start diving into the details.

So, after this presentation, you'll understand the ways in which HTML is used.

You'll understand, appreciate the Single Page Application idea of using HTML And you'll be able to differentiate between HTML and another language called SVG.

So HTML basically the main language for building web pages. It's been around for quite a long time 25, 30 years since the 1990's. Latest version is HTML 5. This complete series of videos looks at HTML 5.

So before we start looking at HTML code, you need to just understand how it gets used. This is the most common way it gets used. Browser sends a request to the server and asks for a web page, okay, such as index.html. Server sends the complete web page back to the browser. Browser receives the complete web page and displays it appropriately.

So that's been going on for like 30 years, that's the basic model and it still goes on today, millions of times a day.

However, things have got a bit more complex in the last 10 or so years.

Let's look at another model, which is Single Page Application (SPA). So now, things are a little bit more complex. Browser sends a request to the server, asking for a small piece of data. It doesn't ask for an entire web page. It just asks for a small piece of data. For example let's say that you had an application for buying and selling stock, stock information, stock prices things like that. So you wanna buy some stock, you send a little piece of information you send a request to the server I want to buy some stock. The server responds with a small piece of data and that comes back to the browser. It comes back to the same, single page application, so you don't get like a flash in a new webpage, you don't get that at all. Most of it stays the same just a little piece changes. So a little piece gets sent back to the browser. And then in here, usually some JavaScript will receive the little piece of information and then update it. Perhaps it is what is the price of some stock, some stock market information. Okay, it just updates that little price shown in the web page. So this is the single page application idea. Requesting for a little piece of data, receiving a little piece of data, changing a small piece of the web page. That is now very very common.

Now, when we say web page, perhaps you think only of HTML, so I'd like to just give a quick example of an alternative, so instead of just doing everything in HTML, you could choose SVG, okay? It's another way of writing a display system, which you give to a browser. Browser makes a nice display, and then you can interact with that display. So it's a simple example here. SVG and then here's some text and here's some other text and the result is this. This is exactly what that code looks like. Is it HTML? No, zero HTML pure SVG. If you want to you could write a complete web system, lots of web pages, and you JavaScript, all of that stuff. Purely using SVG. Totally you could do that. Now here's another example in case you think that's too simple, a bit more complex. Again we're still staying with SVG for this. My web page, we page is so awesome. It's got a nice gradient in the background. It's got a few circles, whatever you want. That is a bit more complex SVG example. And they can do things which is a bit of trouble if you try to do them in HTML like these kind of circles and curves. You could do them in HTML but not an easy situation. SVG is much better for that kind of thing.

All right, so there is alternatives to HTML.

Quick comparison. Well, SVG, very focused on graphics. In fact, the G of SVG, scalable vector graphics, so it tells you it's really focusing on the graphic side of things.

On the negative side, there aren't many libraries with SVG. So for example, you want to do mobile phone applications and so on You wouldn't really choose SVG, okay? You would go for the thing which has been around for like 30 odd years, HTML. Okay, it doesn't have such fantastic graphics as SVG. It really has a kind of fundamentally text approach to what it does. But it has millions of libraries, and you're trying to develop something advanced, you know, mobile phone, multi-tablet kind of system you really don't have too much choice and you really have to go for HTML.

3. Getting to Know HTML

So now let's begin to explore HTML.

So after this presentation you'll be able to build a simple web page and also you'll appreciate the use of an HTML editor.

Here's some of the HTML elements we're gonna look at.

So, first thing is, HTML is defined by a particular group of people called the World Wide Web Consortium.

So there is a particular place on the internet where you can go. And you can see all the information about HTML. And I strongly recommend you don't go there, all right, unless you're kind of an expert. You're really already quite proficient in HTML. All of that detail there, because there's a lot of detail, is really too much. It's not about learning, it's about referring to the HTML standard. So this might be useful later after you build your HTML skills, but right now I would suggest, don't go there.

So lets look at HTML. HTML commands are called elements. Each element has a start tag and an end tag. So this is the paragraph element. There's the start of the paragraph. There's the end of the paragraph. And there's some text which I haven't shown in the middle of the paragraph, okay? That's the usual structure, start, end, something in the middle. However, there are some exceptions which we'll look at later.

So, let's begin to make a webpage. This is the basic structure of a webpage.

You have to have DOCTYPE html at the top, then you start your html element. And down the bottom you finish your html element. And then you have a head section, and then you have a body section. Okay, so this head section is basically some kinds of information which you don't see on the web page. The body section is more things that you actually see on the web page.

So, our first web page. There's the complete source code. We've done the things that we saw on the last slide, and we've added a few extra things. A couple of extra things in the head section. A couple of extra things in the body section as well.

This is what it looks like. My web page. Web page is awesome. Very simple. That is the <h1>. That is the P that we saw, paragraph.

So we saw <h1>, the top thing and we saw P, the paragraph. So <h1> is heading 1. Again, start tag, end tags, some text in the middle. Basic structure of most HTML elements. And also paragraph, start paragraph, end paragraph, and then some text in the middle. Basic commands that you'll see on many web pages.

Now, let's look, for a few slides, let's look at the head part. The stuff that we just talked about, <h1>, and paragraph, was in the body part. What about the head part? So, basically, anything in the head part is information about the webpage. Usually you don't actually see those things in the webpage.

So I had two things in my simple demonstration webpage and one was title and meta tag. Obvious what they do.Title tells the browser what the title is of the webpage. And the meta tag, obviously, tells the browser the author. So, pretty simple, high level information about the web page goes in the head part.

Other things you might find in the head part. You might see style instruction. We'll talk about style a bit later. You might see a link to a style sheet, which is a separate file containing style information.

You might see other types of meta information. Maybe description, key words, maybe character set. Lots of different type of meta information you might see in a web page head section.

Other things you might see, a piece of JavaScript code. That might be in the head. A link to a file which has JavaScript code. JavaScript, we talk about that later. Okay. You might see a base instruction and that's basically saying you have a website, a whole group of web pages. Where are they going? Where do they belong? And it tells the browser where they belong.

So, quite a lot of things that you might see in the head section.

So, we've got lots of elements in HTML. And some of them have attributes, parameters, if you like. So there's one attribute, another attribute for the meta element. So you can use speech marks, double speech marks if you like. And there's double speech marks. But also, if you want, you can use single speech marks. Both of them are fine. The browser doesn't care which one you use.

So, let's begin to learn HTML a bit more. Different ways of learning. One way is to look at the source code of any web page. So here's an example, Wikipedia web page. A website which has lots of general information, encyclopedia information about millions of things around the world. Go to that in your web browser and hit the keyboard shortcut such as CTRL+U on a PC. Or some other shortcuts on a Mac. And you will see the source code. And perhaps we'll have a quick look at that right now.

So for example, this is the website for Wikipedia. I'd like to learn the HTML from this website. So I'm going to go over to my computer and do CTRL+U because this is a PC and just have a quick look at the HTML code.

All right. This is the source code for the Wikipedia website. I can see things I already know about. DOCTYPE html, start html, the head section begins, meta tags, titles. All of these things, a lot of them I know already. Even though, I just began to look at html. But some things I don't know. We can't cover everything in this series of videos. For example, html it can have a language attribute. This particular webpage is in English language. Okay, we don't have time to look at all of these attributes. You can learn them by looking at websites and looking at the source code of different websites. So here's a header section, if I keep going down I'll see the body section. I can learn a lot of things by looking at this kinda website. Okay, another way to learn HTML is to look at different HTML editors. All right, there's hundreds of editors. Many, many to a quick google search and you'll find a big, long list. So they give you a nice, GUI environment, a nice graphical user interface. So that you can build HTML and you can also see the source code that they create. So these editors, they're useful sometimes, especially useful for complementing learning. That's not replacing learning, but complementing your learning of HTML.

Let's look at an example.

So here's an HTML editor, this one is called TinyMCE. And it actually works inside a webpage, because this is actually a webpage. And so let's do a quick example. I'm gonna type some text here in my little GUI environment. And then I'm gonna apply some formatting. I'm gonna apply the <h1> format. All right. So, I'm gonna type some text, it's going to be very small. You won't see it very clearly. And I'm gonna add the <h1> style. And then you'll see it and it a little bit bigger and we can see the source code.

So, let's do some typing. This is my webpage, something like that, extremely small you can't really see it. Let's select it and then apply a format. And we're talking about the <h1> format. And yes, now we can see that we have a large size of text that is using the head <h1> tag. Now what about my source code? I wanna look at the code created by the editor. So I can copy it, paste it into my own system. So let's have a look at the source code. And I can access that with this particular menu. So let's click and look at the source code. And make it a bit larger so you can see it.

And there is my source code. Fantastic, I didn't have to type it myself. <h1>This is my web page.</h1>. And I could copy that, put it in my own web page, put it in my own web system if I like. Of course it's not just the simple things. Like, <h1>. There's a hundred different things you could do with editors and they'll construct the source code and then you copy it and paste it. Does it replace learning? No, it doesn't really replace learning. You still have to understand everything it generates, okay? So that's the end of our first discussion about HTML.

Using a Code Editor For HTML

Lets have a look at a typical editor and how you might use it to explore and develop HTML. For this video, we'll use the Atom Editor, which runs on a Mac. There's hundreds of other editors and on a PC as well, there's many, many other editors. So, this is a typical arrangement. We have an editor on one side. And we have a browser on the other side so we can see the results of what we're creating. So, in our editor we might make some change. Let's change a word such as this. And then save it. We have to save it. And then, over to browser, and do a reload. And then, yes, we can see the change. All right? So, that's how you might do it. Typical pattern, save from editor, load in browser. Now, some editors can do a more helpful job than that. So let's turn that on in the Atom Editor. We're gonna turn on a preview function. So let's turn that on, and then within the editor, it shows the result of the HTML you are developing. So let's do a perhaps copy and paste something, and then we will see the result immediately. We don't have to wait, we do not even have to save it ourselves. It's automatically done by the editor with this preview feature. Very, very useful and can save a lot of time.

Now, for this particular video, that preview window is a little bit small. So we're gonna turn it off and go back to the traditional style of having a separate browser window. Here we are. So, let's talk about these colors for a little bit. So these colors are automatically added by the editor. They are not saved in the source HTML file. And they are just showing you where the HTML elements are. Now this is useful. Let's say we make a mistake or we accidentally do something like this. Well, you can see the next tag has been shown in a different color. It highlights that there's some kind of problem. So, now your eye is drawn to that area and now we could fix the problem. And it goes back to the correct pattern of display. So that's one useful reason for having colors. Now let's look at another feature which some editors have, autocomplete. Let's say you want to have a button in your web page. You type b u and then every element which starts with bu is shown by the editor. In this case it shows button and also some other things as well. So, let's choose it, from the list, we choose button and if you like, we could do a closing it, closing the tag. And as soon as you close the tag, this particular editor will also add the slash button to help you.

Then, all you have to do is add a couple of words in the middle and save the results and go over to the browser to see what it looks like. And then you can see the extended webpage with the button at the bottom. And you can click on it just to check it works. Fantastic. So, this is just a quick video which gives us some quick ideas of why an advanced HTML editor or a general code editor is useful.

4. Some Common HTML Elements

So now let's look at some common HTML elements.

After this presentation, you'll be able to apply headings and sections within your Web page. You'll be able to create different types of lists. And also you'll be able to write comments in your code.

These are the different elements we're gonna look at.

So, quick reminder, what do we already know? Well, we've seen this webpage and we've talked about this webpage.

So let's strengthen our understanding. In that webpage we've already seen, we had h1. So there's actually a range of those h1. There's h2, h3, h4, h5, h6, and each one gets smaller and smaller so that each one is the biggest one. So if you go back twenty, thirty years, the idea of these h1, h2, h3 was a kind of title of the section, okay? So that's why this one is bigger than this one, because it's the main part of the report, for example, and this could be a subpart of the report. All right. So there's different visual size, and going back 30 years, they're supposed to have different meaning as well. Less important, less important, less important. I should say, not less important, but subsection, sub-subsection, and so on. However, these days, people often cheat by using these elements, h1, h2, and so on, simply to get different size text. Now, that is not what you're supposed to do. That's not the correct way to use these elements. However, it is a quick and easy way to get something bigger and smaller. Now, that means that the original idea of these isn't being used correctly. So they added another element called section, which we'll talk about later. So quick example, h1, h2, h3 and a paragraph underneath it. Nothing too complicated. What does it look like? It looks like this. h1, h2, h3 and paragraph. Okay, straightforward. And you can see h1, bigger than h2, bigger than h3, and so on. It works, it looks a bit boring, but it works. It's black and white. Those elements are shown using the default style of the browser. Now, later we'll talk about changing the style so things look more interesting. Let's have a quick look at that.

So this is exactly the same HTML code. h1, h2, h3 and also a paragraph. It looks totally different. That is using a particular set of style rules, which is shared by this particular presentation. But it's still h1, h2, h3, and a paragraph. So we'll look at how to make webpages more visually exciting and interesting later by using style. So, I mentioned before, h1, h2, h3, and so on, they're not really being used in the original way. So recently HTML have added a section idea, so you just say this is a section, and inside the section you can put anything you like, but an obvious thing to do is to put a little title or subtitle, and then some paragraphs or something containing the text in that section. (即用 section)So that's the more common way to handle sections these days. This is what it looks like. In fact, the section command actually doesn't do anything visual, so it's really about structuring the HTML.

So let's look at lists. It's another common thing that you often do in webpages. So, there's two main tags, ul and li. And later we'll look at ol, okay? In this particular slide ul, unordered lists. So we do ul, /ul and then each individual item uses li, list item. So my first item in the list, second item, third item. All of that goes in ul. What does it look like? It looks like this. First item, second item, third item. Very straightforward.

By default, a circle is used on the left-hand side for an unordered list.

What about the other type of list? ol, ordered list. And the list items are exactly the same, no change. We just change the structure to an ordered list, and by default it uses numbers. And so it will automatically increase the numbers 4, 5, 6, 7, if you use an ordered list.

Now, what other things can we do with lists? You could change the start number. Assuming you're using a number sequence. An ordered list. You can actually fix the start number, and then it will increase the number by 1 each time. So here I've said start with 1999. So it starts with 1999 and then each time it adds just to make a series of years just for a kind of funny example.

So we can fix the start number, and also we can do other things. Like we could reverse the list so that it goes backwards. So it's 2002, 2001, 2000 and it goes down. It begins counting down because I added the word reversed. So here it's counting down because of the word reversed. I also, for this particular example, I also swapped the order of those list items.

So what else can you do? Well, maybe you don't want numbers, 1, 2, 3. Maybe you want letters. Maybe you want a capital letter. A, B, C, D. Also, not shown here, maybe you want small letters, small a, okay, you replace capital A with small a, you'll get small a, small b, small c, and so on. So let's use big A, and the result is what you might expect, A, B, C, D, using capital letters.

All right, so nothing too complicated, but they're very common HTML. So the last thing I'm gonna look at is comments. So comments are a little bit hard to type. < !--, that's pretty horrible.

Excuse me. And then we have a comment and then -- >. So, it's pretty horrible to type this stuff, but when you do it, you can put anything you like in the middle. So here for example, This is a simple demonstration of using comments, and I can't believe this, and here's my simple list and so we have one, two, three, we have four comments in one single webpage. Just useful to remind you some things, or you could change some HTML into a comment by adding this at the front and adding that at the end, and it converts some HTML into a comment, maybe temporarily, and it hides it from the browser.

All right, so that's the end of our second discussion about HTML.

5. Formatting HTML Text

All right, now let's look at formatting text inside a web page.

So that's the focus of this presentation and we have several elements to help us do it.

So let's start with simple things; italic. So italic, as you probably know already, you just do italic to some text. Then it will kind of change the angle of the text and make it look a little bit different. So here we have a paragraph, and then inside there, a little italic sequence.

We also, underneath it, have a paragraph, and then we have an <em> sequence. <Em>. Em means emphasis. Okay. So, we got italic, we got emphasis.

What does it look like? Well, it looks like italic and it looks like italic. They look exactly the same. In fact, they are exactly the same. So, this was I / I, italic. This was <em>/ <em>, emphasis. What's the difference? Well, they're both in the HTML 5 standard. Italic is just when you want the words to look like that, emphasis is a little bit more important. They look exactly the same. But one, it could be used for random reasons. And one is for, you should note this, it's more important. That's the idea. The look the same though.

OK, that was italic, right? Let's move on to bold. Something similar with bold. We got a B and a /B. It will give you bold text, thicker text, right? What else do we have? We have strong, /strong. Again, they're both part of HTML 5. What happens if you look at the result? The result is bold text and bold text. They look exactly the same. It's really the same idea as what we just looked at with italic. Okay. Bold, for random reasons you might want that text to be thicker and stand out more using bold. But for strong, it's got a little bit more semantic difference. It's trying to say, pay attention. This is an important message. That's the idea. Doesn't look different on the screen, but it looks different when you look at the source code, especially if you were Google.com doing a search engine, something like that. That will be more important than that. Cuz it will look at the source code. Anyway, two ways to do bold. Let's move on. Underline, so the first thing about underline is don't use it, okay? It's just too confusing. Over the last 25 years, people used it for different reasons and people often think, oh, it's got a line under it, it's a link. They try to click on it. It's not a link at all, it's just got a line under it. Very confusing. So personally, I try to never ever ever use underline. Okay, if you insist on using underline, than this is how you do it. You would do a <u> at the start, </u> at the end. And that will give you a line under it. If you do use it you're supposed to use it only in an appropriate situation. So here's underline, where really these days you shouldn't do that, okay, it really just kind of looks confusing. Is that a link? Should I click on this? I don't know unless I try it, okay?. So I wouldn't do that personally, but if you look at the HTML 5 specification, it says yeah there's some situations where it might be okay. For example, a Chinese name. Perhaps you underlined the surname. Cuz a surname goes first, unlike the western system where the surname goes last. Okay, that is a reasonable use of underline. But even there, people may think wait, is that a link and they try to click on it. So again, personally I would avoid underline if possible.

Okay let's move on. What else can we do? Big and small. You can guess what they do. Big, /big is gonna make this text bigger than the text around it, ok? What else we got? Small, small /small, that text is gonna be smaller than the text around it which is this text. Okay. What does it look like? It looks like this. Okay It's dangerous to use big. Now it may be hard for you to see it, but that word is actually a little bit bigger than the words on the left and the right. Okay. It is bigger. And what about here, smaller, and again it might be a little bit hard for you to see on the video. But those three words are smaller. Okay? Very simple. Big, small. Some things you've got to keep in mind. Although it works right now in this browser, because this is a browser, actually big is not part of HTML 5. It is not. Okay, I'm lucky it works, because luckily we tried it out this year, this browser, it works. However, next year, may not work. It's not part of modern HTML5 standard. Dangerous to use it. It may be three years later, I have a big. Somebody looks at my webpage, totally fails. Maybe it doesn't show anything because it doesn't know what big is. Dangerous to put that in your system, but you could if you like. What's the alternative? The alternative is something we'll look at later called span and a style rule. But we're not gonna look at that here.

So there's our big. There's our small. Let's move on. Highlighting text. Mark. You say mark, /mark, whatever's in the middle comes up as one of those yellow marker pens. You buy them and they go . And then it shows it in a kind of nice thick yellow background. Might be useful for some particular reason. What else do we have? Sub and sup, right? Definitely useful sooner or later. Sub and sup, sub moves the text down. Subscript. Sup, with a P, moves the text up. Superscript. All right, so let me show an example of that. So here's an example. This is a little bit of maths, okay. Now, the only way I can kind of get the 1 to move down and the 2 to move down, the 2 to move up, the 3 to move up, is using sub and sup. Right, that's how I've done it here. I have to say although it works, here's the code, right. Sub 1 that will move my one down next to the x, and then over here sup, over here 3 is going to go up, superscript. Sup /sup moves it up. And over here, sup /sup 2, that's gonna get moved up. So sub, move down, sup moves up. It works, right? It looks like this, yeah I can get some down, down, up, up, I can do some simple maths. However, much better is to use the proper maths language, math ML which we're not going to look at. Math ML will do a much better version. It will make this one smaller because it's supposed to be smaller. It will make the two smaller. It will make the squared smaller. Properly doing maths, you would use maths ML. Anyway, it's a nice example of sub and sup.

More realistic is this kind of example. You have some text and then you use it as a footnote. Again, there's, kind of, other ways to do this, but a simple way to get things working, again you would have a superscript for your 1. And then over here, superscript 1, and then you have your reference. That's a bit better, a bit more comfortable use of superscript than maths, which doesn't work quite so well. Visually, it works, but it's not as good as it could be. All right, that is sub and sup- there's our sub and there's our sup.

What else do we have? Well, there's a couple of things which personally, I haven't really found a strong use for. Inserted text and deleted text. Ins and del. So the idea is that maybe you share documents through a web page and you want to show that somebody else has deleted something, or somebody else has added something. Like you're sharing the document. Alright so insert, /insert, the word truly has been inserted. That is the message you are showing in the browser and also del /del, this word has been deleted. That's the message you are showing in the browser. Looks like this. There's my inserted word. He uses a line to show that it's been inserted, and this word has been deleted. Puts a line right through it. It's really for editors. It's just for editors to show what they're doing in a webpage.

Okay, that's our quick introduction to some of the things you can do with text in HTML 5.

6. Images

Let's have a quick look at using images in webpages.

So, after this presentation you'll have a good understanding of how to do that.

We're going to focus on the image element and these attributes.

So, bringing in a picture into a web page is very easy. You just say image and then the source file name. The browser then grabs the picture and then puts it into the webpage. There is a picture, and this is the webpage here. This white area is a webpage. The picture is correctly shown in the webpage. Now the problem these days is that a lot of pictures from digital cameras, mobile phones are huge, so if you did this then you're gonna get a huge picture in your web page, which usually is not what you want.

So, you need some control, so there is a width attribute and there is a height attribute.

So, here, I've used both. I've used width and height. I fixed the size of the picture to 300 pixels, and the height to 300 pixels. The result is not very good, okay. The picture is too squashed, because the ratio of width to height is not the same as the original picture. So that is a bad result right there. So the trick is to only do one of those. So here, I'm just going to use the width. I'm not going to state the height. Now because I don't state the height that means that the browser automatically works out the height keeping the correct ratio of width to height. So that is much more successful. I just do one of them, browser works out the other one so it looks correct. I can do the same thing using percentage, if I want to. Now, one thing to keep in mind is percentage is relative to the thing that contains the picture. What contains the picture in this example? Well in this example it is the body that contains the picture. So the width in this particular example is 50% of the body of the webpage, and then the height is calculated appropriately.

All right, so there's several things you can do with pictures. We've just had a quick introduction and we'll stop there.

7. Audio

Let's now look at handling audio (au 念哦, 不念噢) sound, in a web page.

That's the whole target of this video. It's just about sound. We need to use the audio element and several attributes.

So, how can we get sound into a web page? Well basically, audio, and then src, and then the name of the file, and then /audio. Okay that's it, that will bring in a sound into the web page.

What's the problem with this example? Well you can't hear anything. It'll bring a sound into the web page. It's in memory, but it won't play it. So how do we play it? All right, we have to add a little bit more. We have to add another attribute, autoplay.

So as soon as the web page loads, it will load the sound file and then it will play the sound file straight away.

So let's check that by running this example over here.

Works fine.

Okay, I'll stop it now and then we'll carry on.

Okay, so.

Oops, let's forget this little thing here. So, basically that was the autoplay. Basically we told the browser, bring in a sound file and then immediately start playing it as soon as the sound file has loaded.

All right, what else could we do? Well, I have no control over the sound file so far. I can't restart it or anything like that. No problem, all the modern browsers have audio controls. All right, so that's another attribute. Bring in the sound file. Show the user some controls so they can play and rewind and fast forward and change volume, the sound. Okay, so that example is here, it should work if I try it right here.

Yep no problem at all. This is provided by the browser okay? That is the controls. I could jump around, jump to the end, change the volume, all of those things. Simple stuff, but I'm just gonna stop it right now and move on to the next example.

Okay what else can we do? We can loop things. Okay we can loop things. Why is that useful? For example, maybe you have a game and it's got some background music. And you only have two minutes of your background music. But the game goes on for ten minutes. Okay. Then you have your game system like this. You say start the system, start the audio and keep looping it. Every two minutes it will start it again, again, again, if the sound is two minutes long, okay? So keep repeating. Might be useful. We have a simple example here.

Loop.

Loop, and its gonna loop and loop and loop for infinity until I stop it.

All right, that was simply the loop attribute right there.

Full screen okay let's carry on.

What about older browsers? Cuz those kind of things we just looked at, they were really for most recent browsers. And sometimes some people don't quite have the most recent browsers. Maybe they're using a mobile phone which is a few years out of date. It cannot be updated, things like that. So, It's wise to do some things kind of specially. One sensible approach is to use MP3 format sound. It's supposed to work in all browsers. It works pretty good on PCs and Macs but in the mobile environment, it does cause some problems still. But still, that is a good target to start with, MP3 format.

And how can you handle these older browsers a bit more professionally? So for example, instead of it just not working, you can actually get those older browsers to actually show a little message. And the way you do it is like this. This could be sound or audio, it could be actually any HTML tag. So you have this tag, whatever the tag is that you're trying to get working. But you know it won't work on some older systems. So you write the name of the tag and that will work for the newer systems. For the older systems, they will ignore that tag because they don't understand it, because they're like three years out of date. So they ignore this and they ignore this. But they do pick up a simple paragraph in the middle. So older browsers, not very good browsers, not very complete, they'll pick up the middle. But the newer browsers will ignore the middle and they'll just use the start and end. So, that's how you handle older browsers. Nothing to do with sound by itself, but actually it's a general method.

All right, let's move on. There's an example. So there's sound, and then for older browsers, or not very good browsers, not very complete browsers, there's a message. That message will not be shown in the newer browsers.

8. Video

Let's now know look at video in a web page.

So, our focus is just on video. We're gonna go with the video element and these attributes. So, we just want to grab a video and bring it into a webpage. We just use the video elements together with source, and that tells the browser where to grab the file and it brings it in, and there's our video, okay? Any problem? Well, just like sound in the previous video, it doesn't actually play the video unless you do a bit more work.

So, just like sound before one of the ways we could play it is auto play. As soon as you open the webpage it will play that video. That's a simple thing to do. Let's try it out. Click on this example and you can see, yeah, as soon as I open the example, it just played it. So, video, handling video is very similar to handling sound. It is the same attributes, it's the same approach. Okay, let's come back to our presentation.

So we've got autoplay, very similar to sound. As soon as it loads in a webpage, automatically (au 念哦, 不念噢) play the video. We have add some GUI controls, again really identical for handling sound. You would do this, the browser has a little panel at the bottom, and then you can control the sounds. Let's quickly do it. So, it looks like this. And zoom in a little bit, there we are. There's our simple panel added by the browser so we can play and forward and backwards on the video. No problem at all. What else do we have? Very similar to sound we have the other things. So let's look at those.

We have loop. It will loop the video again and again. I'm not gonna run an example. It's very obvious. We have alt, alternative text. Very useful for search engines and people who can't see videos, blind people and so on. So, very good idea. You are supposed to add alternative text. All right, what else? Well, we could handle older browsers, maybe old mobile phones, things like that. How do we handle it? Same idea, same way that we handle sound, okay. We have a structure. This middle part gets picked up by older browsers and then the outside part gets picked up by newer browsers and it displays the video. So, handling video very similar to handling sound.

9. Links

Now, let's look at links in webpages.

So at the end of this presentation, you'll be able to add a link to any webpage. You'll be able to add a link to any position within a webpage. And also, a little bit of extra, you'll appreciate the role of lorem ipsum text, which you probably don't know about. So we're gonna look at mainly the <a> attribute. Together with, sorry, mainly the A element together with lots of attributes that work with it.

So links. If you think about it, the main difference between a simple document and the web system that we have. The webpages and everything is links. So you can start jumping around the world to different documents. 30 years ago that was the main contribution, there's always lots of documents. What's the difference between before Internet, and after Internet? Really links is the key point. So how do we add link? Well, we say a on the left-hand side, and then /a on the right-hand side. And in the middle, we have some kind of text. That text acts like an anchor. It's a bit like you have a boat, and then you have an anchor to the bottom. That boat is fixed to a particular position. Similar idea in the world of HTML, we have a, /a. It is fixing a piece of text, usually. And that is fixed to a particular destination. So that makes it into a link, basically. That text, that single word in this example, is now a link. So that's how you build links in web pages. So it looks like this. Now, make sure you run your Gmail account, and then you click on that, and then it will go off to the actual website right there. Simple to do a link. Now, we could do a link, but it doesn't have to be text. The thing which is anchored does not have to be text. The last example, it was text. But here, we're using a picture. There's a first picture, second picture, third picture, and we're using the images as anchors. So there's a nice little picture, it's a picture of the Twitter icon.

Now, if you click on that Twitter icon, it will trigger the link, and it will take you to the website, twitter.com. Same thing here, little picture, PNG format picture, doesn't matter what format it is, a little picture of the Facebook icon that is anchored to a particular destination website, facebook.com. Click on this, off it goes, facebook.com. Click on this, off it goes to Google Plus. And of course, you could do this for a million pictures and a million websites. Nothing special about these. I suppose we could try it out very briefly. Let's try it out. So we'll open this example here.

And there it is. And let's try out one of them. Let's try out the Facebook one. So remember, I did this as a link. This picture is an anchor for facebook.com. So click on this. It'll automatically go to facebook.com. Let's try it out. Click and takes a few seconds, but yes, off it goes to the Facebook website. So pretty simple to do these links in HTML. Let's come back.

For some reason, it's changed the size, but let's ignore that.

So what about if you want to do some positioning inside a web page. Maybe you have a big, long web page and it has section one, section two, section three. You want to jump straight into section three. Not the top. You want to jump right into the middle. So the way to do it is you add an id. So an id is basically a name to some kind of element. You add it to a paragraph, perhaps. And then, you have your link structure, and your link structure uses #, and then the name that you've used, the id name.

So this would be within the document. In this example here, within the same document, when I click on Go here, I wanna jump to this thing, whatever that thing is. It could be a paragraph, header one, header three, whatever it is. It's got a name. I add the name. I add this text, click. It will jump to that position in the same web page. Let's do a quick little demonstration of that. So let's open it up, and what I have here is lots of text.

Also, text to quickly demonstrate it, and I've added the destination right here at the top. And then, right down at the bottom. Let's go down at the bottom, and there's lots of text which we can't read. But that's fine, because I don't want to read the text. I just wanna try out my linking. Lots of text, and then here is my link and within the same web page I can now jump. Let's try out, click on that. Boom, and it goes straight to the top within the same web page. So that works fine. Now, let's just do a couple more things before we finish.

So what about it I want another web page, and I wanna jump right into the middle of this web page? So I'm in another web page, somewhere else on the Internet. No problem. I too, exactly the same idea, that's the location in the web page, I just have to add the exact location of the webpage. Perhaps I'll have to add the server if it's a different server, maybe a sub-directory, but the basic idea, you add the web page source, location, as well as the exact location inside the web page that you want to jump.

And that will jump you right into that location in that web page.

That's the idea of jumping around in different web pages. Now, finally I thought it might be interesting just to look at that funny text that we saw. So that funny text that we saw is called Lorem Ipsum Text. Now, that might be quite useful depending on what kind of application you're building. And it's very useful for temporary text. You don't really wanna read it. It's in Latin. It's got a special name. Lorem Ipsum. It's like 2,000 years old or something, and there's only like two people in the world who can read it, because it's in Latin. But very useful to have that text as temporary text, so you can start jumping around. Fit it in your system, whatever system you're writing. And so, this Lorem Ipsum idea is very, very useful. So what you can find on the websites, on the Internet, is lots of ways to generate that text. You can actually copy it, this is just a random website, nothing special about this one. Google.com, search for Lauren Ipsum, and then this one will actually generate different types of random Latin text. So I say, give me a few paragraphs, few word. Here we are. And I said, build me five paragraphs of Lorem Ipsum text. Great, I can copy this, grab it, put it into my own system as nice temporary text.

The advantage of this type of text is no one can process it. You don't get distracted. Your brain automatically kind of filters it out. So that allows you to concentrate on everything else. Because you're not kind of like reading the news from yesterday, because you can't read it, so it doesn't distract you if you use this text. So that's the general world of Lorem Ipsum text.

10. Void Elements & Breaks

Okay let's look at void elements and some examples of that, particularly breaks.

So after this presentation, which is a very quick presentation, you'll appreciate what a void element is, and also you'll be able to differentiate between three different types of break and also use them.

So, elements we're gonna look at, three different types of break, come back to that in a minute. First make sure we're clear about void elements and what they are. So all of the elements, the HTML elements, we've seen so far, they all this structure here. Basically start tag, and some kind of content and then some kind of end tag, which matches the start tag, okay? So we've seen that many, many times. Lots of video. Now what about void elements? Void elements do not have this middle part. Now because they don't have this middle part that means they don't need an end tag. Right? So, that's gone, that's gone, for these void elements. Also, it doesn't really make sense to call the first tag a start tag, because it's not really starting, because there's no end. So you would just call it a tag, okay. So basically void elements. They just have a tag, and that's it. All right. Pretty simple, pretty straightforward. They must not have an end tag according to the latest HTML specification. Alright, so you only have one thing. An example we've already seen, this one we put this in our header part a few videos ago, and we said, okay, give this information to the browser. The name of the person who made the browser webpage, and so on, and so on. We've seen that before. That is a void element. That's the entire element. There is no slash meta later on. No, we don't use any slash meta. That's the whole thing, it is a void element. All right? Can void elements have attributes? Of course they can, right? There's an attribute there, there's an attribute there. No problem with that. It's simply they do not have a closing tag.

All right, so what other examples have we seen? We've seen <img>, we discussed that before. That's a void element. And a bit later we're gonna look at <input>, when we talk about handling forms. It's a void element. And right now, just to kind of strengthen our understanding a little bit, we're gonna look at three types of break, which are all void elements.

All right, so let's have a look. We've got four paragraphs, right. In this example, four paragraphs. They're all exactly the same text. Exactly the same.

This first paragraph has no special tag at all. Second paragraph has a break tag, BR. Third paragraph has a WBR tag. Four paragraph has an HR tag. All right? Element, I should use the word element. Okay, so we've got one, two, three types of elements being used.

So what does it look like? Let me show you by opening it.

Okay, so this is the example. We've got one, two, three paragraphs we're gonna look at, at the start. First paragraph, nothing special. Second paragraph has a BR, a break. The break is here. Right here, at the end of the first sentence. We told the browser, using BR, we said, okay, go to the next line and then carry on. Okay? So, that's BR and you can see, yes, nothing has occurred after the BR tag, which was here.

Third paragraph is using WBR.

So, what happens with WBR? WBR is a kind of optional break. So, if we look at our first paragraph, we have this huge, long word, and there wasn't enough space to show the huge word, so the browser put this huge word on the next line. It's the best thing it can do. Now what about here, that was just BR, so it does the same thing. That's me forcing a break, nothing special. But here we have something special, WBR tag. So I put a WBR tag right here in the middle of that word. It's a big, long word and I put WBR here. So the browser said okay I don't have enough space for the entire word. So there's WBR, okay no problem. I'm gonna show the rest of the word on the next line because I put the WBR tag in. Without the WBR tag, it would do this, right? It would just put the entire word on the next line, and it leaves a big gap here. Probably not what you like, okay. So that's the idea of WBR and BR is just simple, just forcing the break. Okay, now to illustrate this word, this example a bit more, I'm gonna make the web page a bit wider and you'll see that the entire word will be shown once again. So let's do that. Come over here, make the web page wider like this, and it's put the entire word once again back on the same line because it doesn't need to break it, right? WBR is right in the middle, only gets used when it needs to use it. All right, so there's BR, there's WBR, we've discussed that, and then the final type of break is HR. So HR, it used to mean horizontal rule. So here's my paragraph. Right in the middle of the paragraph I said HR, use the HR element. And so it put a horizontal rule, a kind of line right across the page. Now these days in HTML5, they don't call it horizontal rule, they call it thematic break, which means that you have something and then you're gonna discuss a new type of thing. So, you put a big line there using HR, and then you carry on with a new theme. Okay? Doesn't really matter what it's called, it just makes a line like that, right there, in the middle. Okay, so that's our brief exploration of void elements and three of them.

11. Style

All right, now let's look at style.

So after this presentation, you'll appreciate the concept of style, particularly with HTML. And also you'll know how to create different types of style rule for use with HTML.

So we're gonna look at all these tags and attributes, but let's look at them one by one. And also we're gonna look at some style properties, but let's do that a little bit later. So basics, style, what's it all about? Well, we need style. You gotta have style in your webpage. If you don't, your webpage is extremely boring and people go to some other website, okay? So style is the kind of visual component of a webpage. In the world of the Internet, the way you handle style is through CSS, Cascading Style Sheets.

So the basic concept is like this. You have your information, and the style is really a kind of separate set of information, a separate set of rules, and you add them together and that gives you your visual output. All right, that's the basic idea. So if you wanna think of it in terms of files, you could say that is one file, that is another file, and that is the result you get in the webpage. However, that is a little bit simple.

So we're gonna do style sheets, we're gonna talk about that. Let's say you had a big website and the website has lots of webpages. What you could do is create one single style file, okay? CSS file. That CSS file could be used by the first webpage in your website, the second, the third, the fourth. All of the web pages in your website could use that set of style rules in that file. Okay, now if you do that, that means all of these webpages in your website, all have the same look and feel. They all appear to be the same. They use the same colors, background, spacing, all of those things, so that the website as a whole feels like an integrated entity, okay? So that's a common way to handle the design of a website.

So let's look at the code to do that. Basically, we have our webpage and the things are similar to what we've done before, we have a header, we have a body, the body contains the visual things that you see. But now in our header we add a link to a style file, all right? link href =, and then off it goes, tells the browser to go and grab that file and add the style rules into this webpage.

So, that's what we do. And then these rules in here can use those style rules. Not these rules, these elements, I should say, can use those style rules.

So here's a more precise example. Linking off to a particular file. There we have some particular elements. Nothing very complicated, header one, paragraph, header one, paragraph, nothing very advanced. Now, those, usually they will look like black text on white background, but because we're using some style rules in this file, and let's look at the style rules in that file. Very simple, header 1. Every header 1 in the webpage is gonna be purple text. Every paragraph in the webpage is gonna be blue text. That's the rules we put in that style file. Okay, so the final result, putting them both together, is this. Very simple. We had two header 1s, if you remember the source code, and we have two paragraphs, okay? The two paragraphs are blue, the two header 1s are purple because of our style sheet file, which we had here, okay? Every header 1, purple. Every paragraph, blue, all right? So, it's worked fine, that's the actual result.

All right, now, so we looked at color just now. We set the color of something. What else could we set? Actually, there's hundreds of these parameters, this is just a few. These are commonly used style properties for text, really, okay? And so we've got color for the main text. We've got background for the background color of anything. We've got font family for text font, right? Different types of font. Courier, Arial, all those millions of different types of font. Size, control the size of the text and also where is it going? Left side, right side, middle, all of those things, you would control them through style, the right style rules, okay? So, we saw before an example where we used a style file. An alternative is you don't use a style file. Instead, you write the rules right there in the webpage at the top in the header part. All right, style rules go here, and then your HTML elements go here, and of course, they will use those style rules. So this is an alternative approach. You do not have to put everything into a separate style file.

All right, so this an example, style inside this webpage. There's the style rules, header 1, paragraph, and there's my actual HTML elements. And so header 1, header 1 is gonna come out purple. Paragraph, paragraph is gonna come out blue. It's exactly the same example as before, but now we're putting the style rules at the top of the page.

What's the result? Exactly the same as before. Purple. Blue. No change at all for the visual result.

So, one of the things we could do to use style is use a special ID, a name, if you like, for an HTML element. Okay, so if I said all right, I've got a list item, I'm going to give it a particular name, the name is red. Another list item, the name is orange. The next list item, yellow, and so on. So I could give any element, it doesn't have to be list, any element in a webpage can be given an ID, okay? Don't think this is related to lists, it's not.

Don't think that the name has to be a color. No, that's just this particular example. I'm just illustrating you might want to use some suitable name of an element which reflects how you use style a bit later.

You could use the word, like car, this item has an ID car. This list item has an ID happy happy. This list item has an ID India. Whatever you like, you can use a name for any item.

Now, why do we do that? Well, this is what happens when we don't apply any CSS so far. This is just simple list, nothing clever. The idea, sorry, the ID thing hasn't changed anything. It hasn't added any style, all right? This is what we see so far, and then how do we use our CSS? Well, we would use it this way.

You put the name of the element which you want to apply the style, with a hash in the front of it, okay? So whatever the name of it is, like India or car or green or whatever it is, you put the name there. You put hash in the front, and then you say what visual parameters you want to use, okay? So, it's another way of using style. Based on the name of the element, you can apply a style, all right? So, a particular example here, we've fixed up some names, and given them style for those names, all right? And so these are for those list items that we just saw. And also, one up here, rainbowColors. We're gonna use that for the background of a big list. Okay, so we've got different style rules and then we're gonna use those rules. This is exactly the same as what we saw before, but now we are applying our style rules, okay? So this thing here, red, is gonna use this here. red{ background: red}, that is gonna be applied to that list item, okay? What about rainbowColors? rainbowColors, style rule here. Background color gray. What is that gonna be applied to? And it's gonna be implied to the entire list. The background color of the entire list will be rainbowColors, which is actually just grey. All right, what does it look like? It looks like this, okay? There's the background color of the entire list. There's the style color of the first list item, style of the second one, third one, fourth one, fifth one, sixth one, and so on, okay? Looks very pretty. We have used the background property of these list elements. What exactly what you want to do is up to you, how you use these style rules is up to you. This is just an example just to show some kind of pretty things.

All right, now another way to handle style is to use class, and simply speaking, the idea is you make your own rule and then you apply it to anything. One rule can be used for multiple things.

All right, so here's an example.

Basically you put a full stop at the front of the name of the style, okay? .zappy, and then you define what it's gonna look like. .wow, and then you define what it looks like. Okay, and then how do use it? Well, you have your element and then you say class= and then the name of that thing that you wrote at the top, okay? Header 1 is gonna have the zappy style, okay? What is the zappy style? Okay, it's purple, and yellow background. Okay. Paragraph. It's going to have a wow style. What's that? Oh, that is blue color text and a light grey background. Okay, so some of these have zappy style, some of these have wow style.

All right, what does it look like? It looks like this. It's up to use to choose the colors and how things look, but this is just illustrating our class example. So this one here, and this one here, they both have the zappy style, okay? And then this one here, this one here, they both have the wow style here.

All right. What else could we do? Well, we could define lots of classes and then use them suitably.

So, for example, here's a whole group of classes. zappy, spicy, wow class, lol class, lots of different classes. Doesn't matter what they are. It's your choice. Use any words you like. If you're doing class, just remember the full stop in the front. And then you can use them. Now, what we're showing here is one element can use two different classes at the same time. Okay, so the first paragraph is gonna have zappy style, and it's also gonna have wow style both together at the same time, okay. So it's gonna have blue color, and it's gonna have background color lime color, which is kinda green color, okay? Blue, green. That's gonna be the first paragraph. Let's check the result. And the result looks like this. Blue on top of green, right. So it is correctly using these styles, whatever styles you want, by using this class method, which requires you to put a full stop in front of your name. Okay, so you can have as many of these little classes as you want. Doesn't have to be two. As many as you like. Depending on what you like, and they all get added together. They all get used together. All right, that's the ending of our discussion about style.

12. More on Style

Okay, let's have some more discussion about style.

So basically, after this presentation, you'll understand inline style, and pseudo-classes, and also a little bit about which rule has more priority than other rules.

These are the pseudo-classes we're gonna look at.

So, first thing, inline style. So, inline style is when you type the style that you want to apply right there for that particular element. So for this particular paragraph we want to say all the text will be on the right hand side. That's what we're saying right there. So that is inline style. So that's very different to what we've talked about before. Before, we talked about putting style at the top of a page, and this is really the opposite. This is saying, no, I don't wanna put it on the top of the page. I wanna put it right here for one particular element right there, all right? So it's harder to manage, because you have to go through all the code, and try to work out which inline style is being used, if everything is not at the top in the style section. So it can be a bit more troublesome.

Here's an example. This has a list, unordered list. And we have four list items in the list. And we have a style rule at the top, which says every list item is gonna have a yellow background. Okay. No problem. Four list items, they will all have a yellow background. However, down here, we said, no, I want to apply this inline style to the third list item.

So, because we've said this, this is a higher priority than this. So this one will indeed be in purple and all the others will be in yellow. So we get a result like this. Yellow, yellow, yellow, but then we said inline style for the third list item, so it's purple.

It has a higher priority.

All right. Let's talk about another thing. Context control for style rules. You could have a rule which looks like this. And it says all right, list items, those will be red color. But what it actually says is list items which are within an unordered list, those list items, no other list items, will be red, all right? So, list item inside unordered list, red. Any other list item will not be red.

So here's an example piece of code. We've got a list and another list. This one is an unordered list, this one is an ordered list. And then we've said up here, a rule. So we've said list items in the unordered list, that's the first list, they will be yellow. Now, that will not apply to the second list.

So here's the result. You can see the style rule has not been applied to the second list, but it got applied to the first list.

Let's look at another subject, pseudo-classes. So, pseudo-classes are basically classes with, what I tend to say is, some kind of intelligence, some kind of extra capability.

So here's the first one, and it is hover. So it's saying when the user puts their mouse over heading 1, it will change to red. So, put your mouse over heading 1, the color of the text will become red. Move your mouse away from the heading 1, it will go back to the previous color.

So that's hover, so it's some kind of intelligence, if you like. Now, there's a lot of these pseudo-classes, many, many, of them. So let's just have a look at a few of them.

So for example, here, if you have a link, then that will be a red color, all right? Now that applies to a, because actually, that's what a link is. So it has to be a. Remember a, anchor.

So, any link will be red color. What about here? visited. That means that the person who is using the browser, if they have visited a link and the link is shown on the webpage, then you can fix up a color.

So again, that has to be for a, because a is for links.

So anything that you have visited will be shown in red color.

active. So, active means you are currently following the link. You're currently following the link. Now that could be just like 0.1 of a second. You've clicked on a link, and the browser is just going and grabbing the next webpage and building it, and then it displays the link, it could be very, very quick. Even though it's very quick, there is a pseudo-class for handling that, you're currently following and grabbing the next webpage, because you clicked on a link. All right, it will apply this style. And again, that has to be for a, cuz it's all about links. What about this one? Well, this one doesn't have to be for a. This is pretty straightforward. This is any element which is empty. It has nothing inside it. All right, it could be every paragraph which is empty. It will use this style rule.

So this is just five of them, but there's actually another, 20, 30, 40, a lot more pseudo-classes. So here's an example. And we've set up these rules that we just looked at, the pseudo-classes, and we built a simple webpage to explore them. And then it looks like this, and so we can try some of these out, but some of them, remember, are quite hard to see. So let's go for the hardest one first. Remember, the hardest one is the active link. Okay, that is the link that is currently being processed. It may be hard to see. Purple. So, active link, it will become purple. So let's try to open the example, and then we'll play with it and try these out. Active link will become purple is the first one we'll try.

So here's the example, let's try that first one. Active link should become purple, but you may have to be very quick with your eyes, but let's try it. There's four links here and a list, a series of lists here, one single list with lots of list items. So let's try the active link.

So I'm gonna click on this one, and yes, it managed to show that in the correct style very quickly.

Okay, so that was active link. What else did we have? We had a link which has not yet been activated. You can see there's some yellow color being applied behind it. And what about if we have visited the link? Well, now we have visited the link, so it's using that style rule as well. Down here we have empty style rule being used. All right, so this particular list item had nothing inside it, so that triggered the use of that particular style.

And then the other thing we used was hover, so let's show hover.

So you can see if I put my mouse over these, they change the background color.

Okay, a little bit hard to demonstrate some pseudo-classes because they have some kind of intelligence, if you like. But yes, you can see mouseover has triggered the different style. So that's our brief introduction to three different areas of styles, including this one, which is pseudo-classes.

13. Tables

Let's discuss tables. So very straight forward at the end of this presentation, you'll be able to construct an HTML table.

So HTML tables, they have been around a long time basically, since the start of HTML. And they're the traditional way to get some kind of structured layout in a webpage.

To do this, several different HTML elements work together.

These are the tags we're gonna look at, the elements we're gonna look at. Let's start looking at them one by one. Also we're going to work together with style properties. From the last video, we know that style is the way that we handle different visual elements and it's the same thing with tables. We need to handle these style properties.

So we want to do a table, obviously a table goes in the div area where everything that you see goes. And your basic table structure looks like this, you've got a table, /table. And then inside the table, you've got a header area and you've got a body area. But just to make sure it's not confused with the main header and body area, we have thead and we have tbody. Thead, tbody.

Inside the header part we have a series of cells which are inside a row. So tr is table row. And then over here, finish the table row. And then inside our table row, this is the header column. The very first column that explains the meaning of all the different columns. So we have a single row and then we set up the different cells inside that row. First cell, second cell, third cell. These are the column. You'll see an example in a minute.

That is the header and then we have the body, and this is where the actual main content is. So we could have many, many rows, one row at a time. And each row is usually created using lots of boxes the same number of boxes in each row, usually. Now when I say box, that's what td means. This one single box in the table. Td, table data. So one box there, another box, another box. This has three boxes in each row. Same thing with the header explaining the meaning of each column, they're going to have three boxes as well. Except it's not td here, it's th. Now, if you kinda put it all together you're gonna get an example of something like this. And so there's our header row.

Skills, Difficulty, My Level, so that's three columns being constructed. And then into the main part of the table, the body part. We have three rows in this example. And just like the header, we have three boxes. And that's gonna give us three columns basically. So like we said tr for each single row, td for each box except in the header where we use th, th, th.

What's the visual result of this example? This. There we are nice table. Column one, column two, column three, it's done exactly what we asked it to do. Now, perhaps you're thinking yeah, okay, I can see three columns but there's some kind of well it's all crowded together, I need a bit of spacing, maybe I need some lines to show where all the boxes are, that kind of thing. So that's why we need to move to style so we can handle those things.

So what kind of parameters? Well, lots of parameters. We could use for style. This is just some, there's many others as well. We saw color before.

We saw background before. And here's lots more that we could use. And some of those we'll use now.

So style applied to our table. There's a simple table with just two columns right there in this example. We want to have it a bill better visually, so we know what to do. We add a style section. Of course, you could add an external style sheet file but here, we're putting the style at the top of the file. And then we're saying okay, every table, every table data, little box, and every table header box as well, they will all be shown using these style rules. Padding, which is a little bit of spacing because remember the last example was very crowded. And also we'd like to see a line around all those boxes. So that's called a border and that's how thick the line is and it's a solid line instead of a dotted line. And it's gonna be a black line. Those three are applied to the border property. So that's the first set of rules and then the second one, just for fun. Every single table box in the main part is gonna have purple colored text. You can change it to whatever you like. So here's the result.

Quite nice now, I've got lots of padding. So it's not all crowded together. I've got purple text in the main part, if that's what you want. It's just an example. And I've also got lines everywhere, so I can very clearly see what's happening.

That's much nicer use of a table by adding those file properties.

What else can we do? Well before we talked about classes and so for any kind of HTML, including tables. We could use class rules.

Remember, class rules are when you put a dot in front of an English word. Dot English word, dot English word. And then you have your properties. And that basically sets up a nice English word which you can use. And here I'm using it over here, if you want to. So td class="profit". That's this example here. So I'm saying, okay for this particular box here with the number 200. I'm going to use this particular set of parameters. Light blue background and the text is gonna be on the left side. What else we've got? Another one here. This box here. The class is zero. There's zero. It's gonna use these perimeters and part of this example is to illustrate text align.

Aligning text left. Aligning text to the center. Aligning text to the right. That is part of this example. So I've done that here and the visual result is this one. Just a straight text aligning left, center, right, and also, we changed the background color as well. Many, many different things you can do with styles. I've changed the color of the line, it's green now, and so on. So we've done all those things and sometimes text aligning is quite important, depending on what you're doing. So here we have a more comprehensive example. So I've set up six classes here.

Three of them here are for the vertical alignment. So that's this axis here. So I've said okay here, put things at the top vertically. Here, put things in the middle vertically. Here, put things at the bottom.

And then I've done another three over here. And I've said, okay. In the x axis, put this on the left. Put it on the center, and put it on the right. So I've got six classes there, together with some other stuff as well. And why am I doing that? To illustrate using multiple classes. So the top left cell. The top left box is gonna use the t class and the l class. What's the t class? Vertical alignment top. That means all the text goes right up to the top. That's the first one. This one l or the cell. In the X axis the text is gonna go on the left hand side. That's just the very first box right there. Top left it is gonna put the text whatever the text is. The text is the number 1. Nothing very exciting. It's gonna put it on the top left. So I'm combining two classes together. So I do that with all of these boxes. We have nine boxes. Top left, top center, top right, middle left, middle center, and so on. Just to illustrate multiple classes and also to illustrate different types of alignment, x and y for text. So when we put it all together we get this example here. Just to illustrate those different positions for the text inside td, inside the boxes of the tables.

Now, if you find this troublesome, remember you can always jump off to an HTML editor and they will happily create the HTML for you. So I'm gonna do that right now. I'm gonna pop over here and just use this particular editor and let me just use it properly, insert table, how many? Well, let's do something like this, click. And then he has created the structure for me. That doesn't look very good but that is the structure that I just outlined, a particular number of columns, a particular number of rows. And okay, where's all the lines. Well, we just sort by default, it doesn't show any lines. That's a separate question. But that is the structure right there. And you can start typing stuff in, filling stuff. And then look at the source code the grab the source code and put it in your own system. So let me do that. I'll just type a bit and then look at the source code.

So type of things like over here, like hello, something like that. And then let's check out the source code that I've just done. And we get this. Doesn't look too helpful but actually it is helpful. It has put some non breaking spaces in all those boxes which I didn't ask it to do, but anyway it just helps make it a little bit thicker in the display.

But we do have the things that we're looking at t, table data, table body, table row, we have all of those things there. May be a bit hard for you to see. I'm gonna just sort of try to make it a bit larger and you can see here, those are the things that we've seen already in the last two slides. So if it's kind of too much trouble to make it yourself no problem just use an HTML editor.

And that's the end of our discussion about HTML tables.

14. Div and Span

So let's have a quick look at div and span.

So after this presentation you'll have an appreciation of the role of div and how it could be used. And same thing for span.

So we're just focused on those two elements, and we look at them using a few style properties.

So div has no particular style, and it has no particular meaning. So HTML developers can use it for almost any purpose that they'd like, and it is commonly used by some particular JavaScript libraries.

So let's understand div a little bit. So, we have some HTML. We have a paragraph, div, a paragraph, div. The second div down here has a particular style. I've kept the paragraphs in simply so that they can be used as a comparison.

That's the result, paragraph, div, paragraph, div. This div really has no visual difference to the paragraphs above it and below it. And that's basically what happens. There is no different style unless you apply a style, okay, such as here. Here we have our second div, and it has a style applied. So yes, we can see very much it has a different style.

So, let's strengthen that idea. Let's add a few more style attributes to the first div and a few more to the second div. Basically just playing around, changing size, changing font, changing background color, just for exploration.

So, we get a result like this. Paragraph, div, paragraph, div. And it's just basically playing around with style. All right, now, one interesting thing about the div, and also other elements as well, such as image, is you can position them. And sometimes that's useful. So let's look at that by applying it to div. So, the way to position an element is to add position:absolute together with the top left corner position, okay? Top and left control the top-left corner position of the div, if you're changing the div position.

All right, so for example, if you said top:0, left:0, together with position:absolute, that would fix your element, such as a div, so it is right in the top-left hand corner of the webpage.

So a couple of examples, here we've added position:absolute and then we fix the top-left corner, use top, left and same thing down here, second example position:absolute, fix the top-left corner. So the first one is go down 60 pixels and go to the right 60 pixels. And so we can see that go down 60 pixels, go to the right 60 pixels, that is the top-left corner of the first div. And the second div, something similar. Go down 92 pixels, go to the right 80 pixels. And so 92 pixels, 80 pixels. So you can set the position of divs and other things as well in this way.

Another thing you can do is relative position. And that fixes the position relative to the position it would have been shown if you hadn't changed things.

So here's a simple example, relative position, change basically adjusting the position, go up 20 pixels, go left 20 pixels.

So there's the result, here's my div, and I've said go up 20 pixels, which is why we can't see all of the paragraph text. That's why I included a paragraph, so we can see that it goes above the paragraph, and also it went left. And you can see there is a missing T because it's gone off the webpage, has gone too far left, because that's what I told it to do. All right. That's our quick look at div. What about span? So span is quite similar. Span has no particular style, no particular meaning. But the difference is, you would use it for a few words.

So an example, here's a span, but there's no style being used. There's three words inside the span. Those same three words are being used here together with some style.

What's the visual result? It looks like this. There's our first span, and it looks exactly the same as all the other text, because that's what happens. It doesn't have any default style, but in our second example, I've applied some kind of style, changed the background. And you can see, it's definitely different to the text on the left and the right, okay? So, you do span in this kind of situation, not for a big paragraph, but for when you just have a few words where you want to apply a style.

So that's the end of our quick discussion about div and span.

15. HTML Form Basics

And let's search for something.

Let's search for cats, okay? Using a particular search engine, bing.com from Microsoft. Let's search for cats.

And it goes off to their server, and it says oh somebody has typed in the word cats. And that goes to a particular program. What is the name of the program? Well you can just about see it here, Search. It might be a bit small, but you can see it. On the Bing server there's a program called Search which received the data that I gave it. What data? I typed in the word cat in a form, and then it got sent to it. And you can see it right here, q=cat. It's a little bit small, but it's there. q=equals cat. q means query, right? The query that you have is cats. Give me lots of web pages all about cats. And there's some other things as well, I didn't type those in, but they're just kind of added by Bing. The ones we're interested in is the first part right here.

Okay that is the get method. So I sent data from a form. I pressed Enter. It was given to the Bing server, and that was using the get method. Whenever you use the get method, you can see exactly what you type. You can see it. And that's an example. Any problem with that? Well usually there's no problem, seeing it. In fact it's quite helpful. If you're building a system and you've got a browser, you've got a web page, you've got a server program. It makes a lot of sense to use the get method. Because you can see exactly what's happening. Okay, that's the get method. Let's go back to our presentation and look at the alternative.

Okay so we had a quick discussion of the get method by using bing.com. Let's now go and have a quick look at the post method. As I mentioned project you're developing get method, very useful. You can see things. You can see things in the URL. URL means that big web page link thing, okay? However, you can't keep any secrets. And I have an example of that in a minute. The other thing that you have to keep in mind, the get method is only for a few hundred letters or characters. If you type a huge book and you try to send it to a server, it's not gonna work. This can only work for a few hundred letters or characters that you type.

You want a huge book, you wanna send that to the server, you gotta go to the second type. You gotta go to the post method, all right? Post method, you can send huge files. You can send videos, images, all of that kind of stuff, if you just say method is post.

Also better for keeping secrets. Why? Because none of that data is shown in the URL at the top, none of it. None of the things you do using the post method is shown. It's kind of secret.

All right, so let's think a bit more detail. What can we put in our form, cuz we've talked about kinda the high level details. Lets see some exact example. So, very simple form here, simple paragraph. You can put any of these things inside a form paragraph, divs, header 1, header 2. All of those stuff, no problem, mix it all together. Now, this is the thing which we're focusing on, textarea. So, you have to say, start textarea, end textarea. Maybe you say how many rows, how many columns, and you give it a name and we need that name. So the name is passed to the server program. And then the server program says, oh right. Yes, they typed this. And that was the feedback so we've got to give it a name. Let's see what it looks like. It looks like this, all right? This is some default text which I told it to do. Where did I tell it to do it? Here, this is the default text. But I could delete it if I want and press a Submit button, which I haven't added yet, and it will get sent to the server. So this is a text box, very simple stuff, type stuff in, whatever you like. Or I could delete the stuff that's already there. You can say, hi I am Dave, whatever you like, whatever . All right, and whatever you type will get sent to the server when you press Submit. So let's add a Submit button. It's the obvious thing that's missing. So let's go to our next slide.

And we've made the example much more realistic. There's our text area from the last slide. There's our Submit button. Because we've gotta send it, so have to have a button.

And also I filled in some of these essential parameters. What type of method? Well, we're gonna use get method cuz it's great for getting started. And where is it going? It's going to a particular program I have on my server, all right? This is a real location, there's a server, sub-directory, and then there's a particular program written in the PHP language. Doesn't really matter what language it's written in, but it knows how to receive data from a form and then it will do something with the data, okay? So it's a real world example. And let's open that and try it out.

Okay, so it's exactly the same as before, text box and this time we've added a Send button. All right, so whatever text I put in here is gonna get sent to the server. Let's quickly throw in a few things.

All right, there's a few things, just so we can see it when we look at the server result, and I'm gonna hit Send.

And what happens, it just looks kinda crazy, and horrible, and nasty. No mistake, it has totally worked correctly. So I typed in some text and that text got sent to the server, okay? I told my program on the server to take any data it receives and then show it on the screen. If I told my little program on the server, which is a PHP program, I said, display every single piece of information you know about because that's helpful for learning. So that little program on the server, it has received the data from the form and it's printed out every single bit of information it knows. I'm gonna make this bigger and so we can zoom in on one part.

So zoom, zoom, zoom.

All right so this all the data from my particular server program and lots and lots of data and I don't really need all of this data. The bit that I do need is the information which was sent to it. Where's the information sent to it? And here we can see it, okay? You can see it in REQUEST_URI. REQUEST_URI has the data. One, two, threeeee, that's what I typed in. I typed in one, two, threeeee, I pressed Submit. It got given to the server program. And then I told my little server program to display all the knowledge it has. A lot of the knowledge is rubbish, but the the stuff we are interested in is the thing that I typed in, one two threeeee. It has received it. And also it printed it and back that comes to the browser, all right? So, I have got the data over on my server. In fact, I've got it in two different pieces of data. One is this one. Your REQUEST_URI. It has the things I typed in, and there's also one here. QUERY_STRING. One, two, threeeee. That's the same thing, that's what I typed in, so it actually is in two different variables on the server program. Now the server program, we're not gonna look at it in detail right now. Come back to that later in some other videos.

So that's the end of our presentation and we'll finish there for this introduction to HTML forms.

16. More on Forms

All right, let's look at forms in a bit more detail.

So, after this presentation you'll be able to differentiate between quite a lot of the many different types of form input.

These are different elements we're gonna look at.

And also a lot of the attributes which work together with the elements.

So, where are we at the moment? Well, we know this is a typical form. We have an action. We have a method. We have some kind of input, such as a text area, and we have another important element, which is the submit button. That's a typical kind of form right there.

What else can we do? Well we can do all these extra things.

So, this slide shows a text input type, checkbox input type, and radio input type.

This checkbox and radio have a group of different inputs which all work together. So, they all need the same name. That's the name which gets received by the server, okay? Same thing over here, radio, they need the same name. What does it look like? Okay, checkbox, radio. This is our checkbox area, this is our radio area. And to explore this I'm going to have to jump off to a particular web page in the browser. So, let's come over here. It looks like this. And this is our check box. Behaves like this you can have any kind of combination of choices that you like. None, all whatever you want. And then down here we have the opposite idea. The opposite idea you're just choosing one item instead of choosing Any number of items. And up here's just our simple little text, which is just for one small piece of text instead of a large area.

Okay, nothing too surprising and let's just carry on.

Okay. So, that was our straightforward simple inputs, there.

What else could we do? Well, there's a password-type of input and that's quite useful. Often you have to log in. Right? Log into Facebook or something. You have to type your password. It's a piece of text.

You don't want it to be shown in the form. So, there's a nice password input control. What does it look like? It looks like this. It's just a piece of text. Nothing too surprising but it hides your text let's play with it and type in some kind of secret password. Whatever you want. La la la, la la la la. Great, it doesn't show it. Everything is fine, no problem at all.

However, there is a little problem if you're not careful.

So, let's go to the next slide and let's think how you might do this There's your password input, submit button. Here's all the details. Everything looks fine. No problem at all. However, there is a big problem. The big problem is if you actually run it. So, perhaps I'll just run it right now.

If you run it then, you're gonna get a big problem if you use the get method. So, let's do this. Something like this. Type in something, some kind of, whatever it is. And let's send it to the server. Looks scary, but remember, this is just my test server program, which dumps out everything it knows about and then sends it back to the browser. All right, so, I passed in a secret password, but look, right up here, because it uses the get method, I've got my secret password right there. My secret password is secretttttt with lots of ts. That's what I typed in. I typed it in again sent but by default the browser uses the get method, the get method will show everything you type in. Total disaster no secret at all, okay. It works, I mean it sends it to the server but you know, people could read this very easily. Not good solution at all. So, gotta be very careful if you use that password input.

Okay, some of the other things you might like to do with the form, you might like to select from a list, a particular list, and the option and select combination, is what to do that So, this is a drop-down menu, and it will show all of those things that are set up in the form. So, if I click on here, it looks a little bit small because I have a kind of zoomed in web page, but actually, it's correctly showing Those different things I asked you to list. It's not showing it very large, but it is showing it, okay? So, I can choose one of those things, press the submit button which I haven't shown here and it sends the selection over to the server.

Okay. Plenty of other things that you can do with forms as well.

So, for example, you can basically add some kinda preferences, whatever you want. So, if you want to show something in a text box, right at the start before anybody types anything. You can do that by using a value attributes all right? If you want to show some text right at the start before anybody types in anything. But you want the text to disappear when they start typing, then use placeholder. You can use autoFocus which means the browser jumps to that field. And highlight it and encourages the user to do that field first, okay. And so I suppose its a more important field. Also, you can say user must do these fields and cannot submit the form unless they do these fields. These are attributes you can add to almost any of those form inputs that we've seen So here's an example of the things we've looked at so far. I've just loaded this example. We have three fields. First name, Last name, Age, and a Submit button.

Now, we're showing several attributes here. The first one is Autofocus. And we've used autofocus to tell the browser to highlight a particular field, which happens to be the first field, as soon as the example is loaded. So, you can see that by this kind of blue border around the first field. Second thing is the use of value and I used value so that D-a-v-e is automatically shown in this field. I didn't type it, I haven't typed anything yet. But by using value, it automatically shows it when I load it. I could click in the field, and I could change it, edit it, do whatever I want, but at least it shows me some starting text.

Okay, in our next field, we have placeholder. So, this is placeholder, and you can see a big long message here. Your last name goes here. So, I could start typing something, like whatever I want, and as soon as I start typing, the text disappears. So, that's the difference with value. By using placeholder I can do a message, it completely disappears when I start typing, all right, down here we have a example of required. So, this is our third field and it is required, which means if I don't type anything and I press submit Up comes a message. It hasn't sent anything to the server. Message comes up please fill in this field because this is a required field.

So, I'd have to fill something in just type something and then I'd be allowed to submit.

All right couple more things, these are labels, because they are set up properly if I click on the label it will highlight the field which it is for. The first label is for the first field. The second label is for the second field. When I click on the label, Highlighted the relevant field, the third label, not surprisingly that is for the third field. So, it's just a nice little feature you click and it highlights where you have to enter data. Okay, that's using label and for, all right let's continue So here's the code of the example we wee just looking at. Here is our first label and first input and second label, second input. Third label third input. Now, in our first label and input we were illustrating auto focus. That shows a blue box although it may be a slightly different box depending on the browser around this particular field. I was also illustrating value and that tells the browser to put some kind of text into that box straight away when the code is loaded.

In our second example, we were illustrating placeholder. There's the text which was automatically shown and that text disappears as soon as you start typing. And in the third example, we were illustrating required. And so, if you remember, we could not submit anything to the server until this particular input field was filled in. It doesn't have to be one input field, by the way. It could be all input fields. If that's what you want. So, those are the main things we were looking at and also label together with four. So, label just means some text so we have our text and that text is mapped to a particular input field. Which one? Well, it's mapped to first name That is what we see down here. The id of this particular input field is first name. So, that matches what the label is for so those two are a pair, right there.

Now, going to be a little bit careful, I've also used name over here, and, I've used actually the same text inside it, first name, so what's name for, so name is useful on the server when this set of data is sent to the server, it goes to a program, the program will say, okay first name equals Dave if you type Dave, or David or whatever you type It will be mapped to first name. And that comes from the name. That's a bit different to the ID. So, the ID is useful for programming, but also, in this case, useful for saying which field the label is for.

Okay. So we have the same pattern down here. The label is for a particular ID. And same thing here, the label is for a particular ID. And I just chose to use the same, Text for the name field because that makes it a bit easier for server side programming, but that's not a requirement.

Okay so we've had a good look here at our introduction to forms and that's where we'll finish our more on forms presentation.

17. Handling File Upload

All right, let's look at handling file upload from the form point of view.

So basically, it's enough skills after this presentation so that you can actually handle some kind of file upload system, but we don't really look at the server side of things.

So, this is all related to one particular attribute, file type, input type is file. So, there's actually two sides when you want to build a system which will upload a file and then send it to a server program, and then the server program has a copy of the file. Of course, you have to handle the form side of things, usually, and also the server side, program side of things. Now, this presentation is pretty quick. We're just focused on the form side of things rather than the server side.

So, let's have a look at a basic structure. We know that we have form, we know that we have some kind of action, some kind of method. Something we haven't seen before, encode type. That tells the browser when we send the file to attach, let's use the word attach, to attach the file, or files, that you want to upload to the message which goes to the server.

All right? So that's essential if you're doing file upload, all right? If you're not doing file upload, you can kind of forget about it.

The other thing is this, method is post. Well, we had a brief discussion of that before. If you're doing big things like files, absolutely essential you use the post method. Use the get method, it totally won't work. Use the post method and it should work, okay? So that's some extra things you have to think about.

You also have to do this kind of thing here. This gives you a file selector. So the user sees the form, sees the file selector, click on it, select the file in the hard disk, and then press submit button, off goes the file to the server program.

Here's a real world example where I've filled in the parameters. So yes, post method, and there's an exact URL.

And the other stuff is pretty much the same. Some kind of submit button, some kind of file selector button, okay? So that's a real world example. I won't do it right now. The reason is, I haven't actually handled the server program in this example, simple example.

To discuss that in detail is another presentation. So, we're not gonna discuss the server program in detail now.

On the form side of things, the HTML, this is what it looks like. This was my file selector. This is the submit button, of course. And that's all we need, right? Do those have the parameters correct in the form at the top and then you press Upload and the file will go over to the server.

What does the server do? Well, the server, program on the server, as we briefly mentioned before, it could do a million things. For files, it may have some extra things to do. The server program has access to the file sent by the browser. But it may have to, for example, move the file to another directory. It may have to change the permissions of the file. It may have to do quite a lot of different things, save the file in a database. It totally depends on the system that you're building, okay? But in this presentation, we're just focused on the HTML part of handling file upload.

18. Some New HTML5 Input Elements

Let's have a quick look at some new HTML5 input elements which might be useful for your first assignment.

So that's our focus. The elements we're gonna look at are these, and it's pretty obvious what they're for. Basically, we've got an input element for handling numbers. One for handling dates, one for time, one for colors, and for selecting from a range.

So I've put all those in this particular example, so let's run this example.

So the first one is number so it's not too surprising you can type in a number, whatever you want, and then you have a nice increase and a decrease. Now what happens if you try and type in some letters. It gets rejected, which makes a lot of sense.

Let's move on to the second type, this one is a date type. So you can enter some kind of date, like 9th of December, whatever, 2018, nice and simple. And, if you try and type in some letters, then it gets rejected, only digits are accepted, as you would expect.

You also have a kind of up and down thing here for each column. So basic input there over date. There is also a nice calendar feature although this depends on which browser you're using so you can select which month and which day and which year and that kind of thing and it gets into the main display.

Let's look at the time input. Well, nothing too surprising, enter some time, 9:30 and if I press the a key, the a letter it will show AM. If I press the p key it will show PM and if I try and enter some letters here or here it gets rejected. That's nice and we can do increase and decrease as you might expect. So the full form is color input so let's choose a nice color. Let's choose, we could choose over here or we could choose here. Let's choose yellow and do OK. And depending on the browser it will show the color right there. So it stored the color yellow and that will be sent to the server later when I press Submit.

The last one is a slider so it's a simple selection from a range. This particular example, I have a word over here and I have a word over here. So it looks like I am selecting from a range, but that is not true. I am actually selecting from a number range. Nothing to do with the words, okay? So those are the five inputs and as usual if you press submit all of that data is sent to the server, and then on the server side you can extract the perimeters and do things with that information. Let's have a look at the code for this example.

Looks like this. So there's our input types, number, date, time, color. And in the final one was range.

And as you might expect, you often have to add extra parameters in order to set things up properly. So for example, in this range example, I set up the minimum value, the maximum value, what the increase is as you drag the slider, and also what the starting value is. So you have to set up all of those kind of things.

I've also set up labels for all of these input controls. As we know we use four and then that will set up a label for a particular item which has a particular ID.

And the name is the key thing when the data is sent to the server.

Okay, so that's the end of introduction to five new HTML elements.

19. Element Grouping

Let's look at element grouping in HTML.

So basically, this is just very simple presentation talking about how to group elements together into useful groups. Two elements we're gonna look at, fieldset and legend.

So the idea is nothing too complicated, but it's simply to put items that work together in a similar structure in the same structure. There's two ways to think about this. One is the visual output on the webpage, and then the other is the code. So, the code, you might want to group things together. And the way to do that is to use fieldset together with legend, and when you look at your code, you can realize, oh, yes, I understand, these inputs are related to personal information. So it helps you as a programmer. But also, there's a visual result, which helps group this together as well. So there's a visual side, there's also a kind of programming side, advantages to using fieldset and legend. So, here, I've got my fieldset, I've got my legend. Personal information, two inputs. What about here? Another grouping, fieldset, legend, Favorite things, okay? So I've got two groups which work together, all right? What does it look like? It looks like this. So there's fieldset and there's the legend. This is fieldset has resulted in the browser adding a box around those things, which are put together in the fieldset. Okay, so I had a couple of things. They're put together. There's another couple of things. They're put together. It's a nice visual identity, as well as, in the HTML, a nice kind of coding identity as well, grouping things together. So that looks good. This example is using a form. But it doesn't have to be a form. So, let's have a look at an example which has nothing to do with forms. This is just some kind of text output in the web page, right? A couple of things here. A couple of things here. I want it to be shown with a nice box around it and a nice title at the top. Nothing to do with forms. Okay, I use the same code, and we get a nice little summary, perhaps this is for a game that you're building, something like that. Enemies, friends, this doesn't have to take up the whole page. It could be over on the left hand side. Or on the right hand side. Maybe the bottom, the top, you could put it in a div and put the div at any location, as we talked about it another video. So it's a nice visual way to group elements together.

Further Resources

Having completing this module, you have a good understanding of many common HTML elements and CSS style rules. However, only the most common HTML elements and CSS properties have been discussed.

If you want to learn more about the many different HTML elements, you can find a good reference list of all HTML elements here: W3School HTML Element Reference

If you want to learn more CSS properties and usage, there is a good reference site for CSS here: W3School CSS Reference