**Формат html документа.**

Every HTML page should start with the doc type or document type declaration. The words doctype or HTML could be lower or upper case. In the past, these declaration were pretty complicated looking, certainly not too many people would be able to type them up without copy and paste. HTML 5 however changed all that.

Now the declaration is as simple as it can be. All it does is tell the browser that it should get ready to render HTML. Now I know what some of you might be thinking.

What else would there be if not HTML? There doesn't seem to be any practical purpose for this declaration. If you're thinking that, you're absolute right. This declaration is really largely historical. When HTML standards were first becoming popular, the web was full of pages that were not compliant with the standards. To help browsers render those pages correctly, browsers used the doctype declaration to distinguish between noncompliant and compliant pages. Noncompliant pages were rendered in what's called the quirks mode,

and the compliant pages were rendered in what's called the standards mode.

Now, that's all historical.

But what you need to know today is that if you leave off the HTML page declaration,

that will signal to the browser that it should treat your pages as one not following HTML standard.

Next, goes the html tag, and that's basically a tag that contains the entire html document.

After the html tag, goes the head tag. The head tag contains items that describe the main content of the page. Things like what character coding should the browser use for the main content. It can contain authors description of the page, page title, and whatever other external resources are needed to render the page properly, among other things. The point is it contains some metadata about the main content.

After the head tag goes the body tag. The body tag is the root of all content that is visible to the user. It is often referred to as a viewport. We can now write our content.

**Content models**

All elements fall into basically two categories under the traditional content model structure. Either block level elements or inline elements.

Block level elements render to begin on the new line by default. Every time you specify a block-level element in HTML, the browser will automatically place that element on a new line in the flow of the document. Block-level elements are allowed to contain inline or other block-level elements within them.

This is in contrast to inline elements, which render on the same line by default. Again you can change that, but by default it renders on the same line. Which means if you put a whole bunch of in line elements next to each other, they will all be going on the same line, as if there is no new line character present. Inline elements also have a restriction that they can only contain other in line elements. In other words, an inline element cannot have as part of its content a block level element.

**Headings**

Headings are trying to convey is that the heading content between opening h1 and the closing h1 element is the most important heading in the document.

So h6 would be also a heading of the document, but it's the least important one of them all.

Browser defaulted to styling that basically, visually tells us that this is the main heading.

Couple of important points to understand about these elements.

First, even though their default rendering in the browser appears to give them visual distinction, these should not be used for styling. These elements are only meant to convey structure of your HTML page, nothing more. With CSS, any regular development can be styled to look like any one of these heading tags.

So, why not just use a div? Because if we did, we would lose the meaning of what a heading is.

Second, something that's marked h1 is obviously the most important and generalized description of the content of this page. And while there are disagreements among the SEO experts about how effective these semantic tags are for helping your search engine rank in general, everyone agrees that when it comes to the heading tags, and specifically the h1 tag, that it is of utmost importance to use it and that it should contain the wording, which truly conveys the central topic of the rest of the content.

The header tag basically contains some header information about the page. So usually, it consists of company logo, some tagline, sometimes, navigation. Actually, often, navigation is contained within the header as well. As you can see, this is exactly what we have right here. The nav tag signifies some content that is used for navigation within our website.

And then we have a couple of set of section tags here. And within each section tag, we have a bunch of article elements. If you look at the HTML5 specification, you'll see that the way the section element is defined and the way the article element is defined is that it's usual or it makes sense that the article should go inside the section element.

An aside tag is basically an element that communicates that there's something that is inside of this element that is related to the main content of the page, but not as direct a relationship as the main content.

And finally, we have the footer tag, which, just like it sounds, has the footer information in it.

Now, the thing to note about all of these tags is they're all block level elements. So as far as we're concerned, visually we might as well have just used the div tags everywhere.

**Lists**

So if you look at the page right now, the way it looks in the browser, is you can see there is no list. It's just a bunch of spaces in between them. And the reason that is, as I've told you before, all the different new line characters or any other separator characters such as tabs, extra spaces, they're really all ignored by the browser and only one space is honored.

And since typically, our shopping list is not something we need to buy in order we can use an unordered list for this example.So the way you create an unordered list is you specify a ul unordered list ul tag around your entire content, and every item within the list Is in li tag, otherwise known as list items. Text not allowed inside ul In this context. And what it's telling you is that the only thing is allowed inside a ul element is an li element. Anything else is not allowed.

So as you can see in terms of our html, the only difference is that our ul tag got replaced with an ol tag, which stands for ordered list. But the li items are exactly the same.

**Entity references**

Since HTML uses certain characters for its syntax, we need a way to differentiate between those characters as HTML and those same characters as content. If we want the browser to interpret special HTML characters as regular content, we need a way to escape them.

Specifically, there are three characters that should always be escaped to make sure they don't cause rendering issues, either right away or down the line. And these characters are the following. The < character, the > character and the &. Instead of using the < character, you should use the HTML entity which starts with & and then followed by lt;.So if you put in your HTML, &lt;, what the browser will interpret this as a < character. And similarly, for the > character, it's &gt;. And for the & it is &amp;.

In reality, HTML contains a whole ton of different HTML entity references. One particularly common one is the copyright symbol. And the copyright entity reference is just &copy;.

There's another HTML entity reference that is very commonly used and unfortunately, a lot of time misused as well. Let's say for example that in the last sentence here that ends timid souls who neither know victory nor defeat. Well, the way you do that is with a non-breaking space. And the way you use it is nbsp;. Not breaking space, or removing all spaces between the words.

**Links**

Links pretty much what makes the web what it is.

So the first type of links that we're going to take a look at are internal links. Here I am looking at the file called links-internal.html, and it's located in the examples/Lecture09 folder. And here we have a couple of links we're going to take a look at. Now the way you create links is by specifying an a element with an attribute href. And href stands for hypertext reference. The value of href can either be a relative or an absolute URL. In our case here, since we're discussing internal links which are links that point to internal web pages of the where application, the links we're showing are all relative URL links.

However we could still have an even absolute URL links as long as they point, again, to the same web application that we're in.

It's also a very good idea to always specify a title attribute for the a tag. The title attribute is used by the screen readers that help the visually impaired people get through the web page. The content in between the opening and closing a tags is the content you're going to see in the web page, when it is served and on which you're going to be able to click to go to the href that the a tag is configured with.

If you take a look closely, the a tag is both a flow content and a phrasing content. In other words, to map it back to the HTML four days, the a tag in the HTML5 is both an inline element and a block level element at the same time. And this is what allows us to take the a tag and surround a div tag inside of it. The authors of the HTML5 specification realized that there are a lot of times where you would like to be able to click on a whole region.

Next, let's take a look at an example of an external link.There's really nothing super special about external links other that their href value usually starts with http:// because it's usually the case that external links are hosted on a different domain name than your website.

However, there's one feature of the a element that I would like to demonstrate that is quite often used in conjunction with external links. And that is the target attribute. Target attribute, when it's set to the value \_blank, forces the browser to open this page in a new tab or a new window.

Another type of link that is extremely important to know about is a fragment identifier. So here I'm in again in Sublime Text in links-same-page.html. It's a # followed by some name like section1, section2, and so on.

Now what these links are pointing to is a section of our page. Now you could identify a section a couple of different ways. You can have in any tag that has an id with that section name. Notice that the section name does not contain the # sign. Only the link to that section contains the # sign. That's one way to identify a section within the page. Another way is, if we scroll down all the way to the bottom of the page, is to create an anchor tag with a name attribute and name the section very similarly to the way you name a section id. The way you refer to these sections is exactly the same, you put a # in front of the name of the section and stick that value in the href attribute of an anchor tag.

**Images**

The image is displayed in HTML using this img tag which is just short for image. And the src attribute of the img tag is a URL that points to some image file. Now this URL is no different than the href that you've seen with the regular link A tag. It could be a relative URL, or it can be an absolute URL, even one that contains an external link.

Next, we specifiy the width and the height of the image. While not absolutely required, it's a really good idea to always specify the width and the height.

An img tag is an inline element. If it weren't, the image would be on its own line, pushing everything behind it onto a new line.