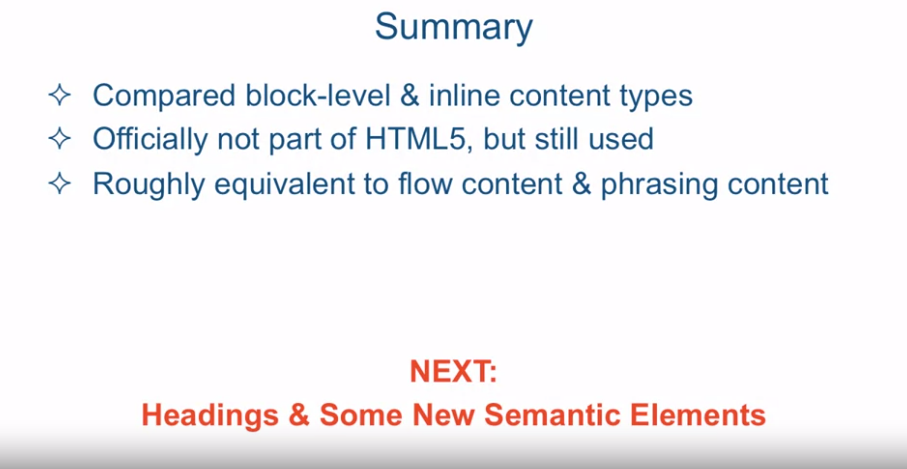
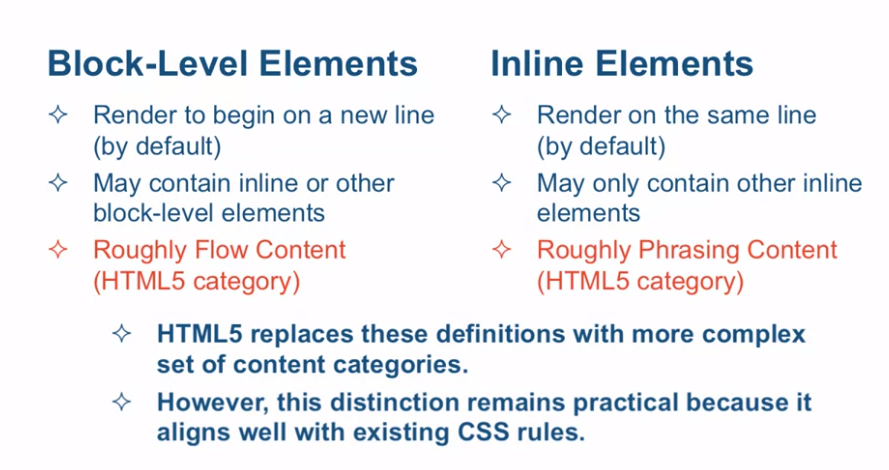
Links:

1. <https://www.w3.org/TR/2011/WD-html5-20110525/content-models.html>

Content model: the full behavior the browser applies to the elements belonging to that content model, and to the nesting rules of those elements. In other words: which elements are allowed to be nested inside which other elements.

Before HTML5 were block-elements or inline-elements, html5 splits these two content models in seven models.





Div stands for division, span for span. The div element is the most generic block level element, while the span one is the most generic inline element.

[SOUND] The term content model refers

to the full behavior the browser applies to the elements

belonging to that content model, and to the nesting rules

of those elements. In other words, which elements are allowed

to be nested inside which other elements. Prior to HTML5 specification,

HTML elements were either block level or inline elements. HTML5 split these two content

models into seven models. So, things got a bit more complicated. Let me go over the two

traditional models anyway. And I will explain in

a moment why I am doing that. 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. You could change that with CSS but we're

not talking about CSS at this point, yet. So what that means is 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 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. Now I told you that HTML5

really replaces these definitions with a more complex

set of content categories. So why are we going over them now? Well the reason we're going over

them now is because this distinction between block level elements and

inline elements remains pretty practical. Because it aligns very well

with still existing CSS rules. So even though HTML5 came up

with new content model names and new sub categories, and

just new way to break them down. At the end of the day,

you could still look at all of these, as far as your coding is concerned, as

block-level elements and inline elements. Obviously, it's a little bit of

an over-simplification, but it works. And just to be kind of complete,

block-level elements roughly translate into the new HTML5 category of

flow content, and inline elements roughly translate into the HTML5

category of phrasing content. So let's go look at some code that

will demonstrate these concepts. Demonstrate the difference between

block-level elements and inline elements, we're going to take look at an HTML

document called div and span the HTML. It's look Located in

the examples Lecture05 folder. Perhaps the most generic elements

in each category are the div and the span elements. And these are the elements we're

going to introduce in this document. The div element stands for division,

and the span element stands for span. The div element is your most

generic block-level element, and the span is your super generic,

inline element. So let's go over this document real quick. We have a couple of divs following

one after the other, DIV 1 and DIV 2. Then, there's a span

element that follows DIV 2. And the number 3 div is a little

bit more complex in that it includes a span element inside of it. So let's take a look at what

this looks like in a browser. So you can see the DIV 1 element

is all by itself on its own line. And so is DIV 2 element,

all by itself on its own line. Now the Span 1 was a tag, an element,

that followed directly after DIV 2. And even though span is an inline element, since DIV 2 requires that

it be on it's own line. It pushes the next inline element

to it's own line as well. And this is exactly what

happens with DIV 3. Even though span is an inline element, technically speaking the tags shouldn't

go anywhere but right behind SPAN 1. But since DIV 3 is a block level

element it requires it's own line so it's get pushed to the next

line to be by itself. Now the spin two tag is sitting

inside the div three and since it's an inline element it

doesn't cause any more formatting And it just sits right here inside the DIV

3 without requiring a new line. So just there's no confusion, the new

line characters that follow the div tags make absolutely no difference to

the html page and how it renders. I could remove all the new line

characters, save the document, and preview it in the browser. Again, and as you could see

in terms of formatting, and in terms of requiring new lines,

nothing gets changed. As the last step,

let's try to take the code from our page, copy it, and

validate it in the W3C validator. As you can see, the page is valid. But what happens if I go ahead and,

right inside our validator, add another div tag right inside

the span tag with some content. And I'll close the div tag,

and let's check the page. And now you can see it's complaining that

element div not allowed is a child of span element and it's context, and it's telling

you the context in which div element might be used is flow content and content model

of span element is phrasing content. And again phrasing content roughly

translates into inline content. And flow content roughly translates

into block level content. I'll provide a couple of URL's for

you to explore a little bit further into different categories of

the HTML5 content models. But I'd like to demo one real quick, and

that is the W3C kinds of content section, where it basically lists the seven

types of content that HTML5 defines. What's cool about this page is that

you could roll over different parts of this graphic and

you could see all the different elements that belong to this particular

HTML5 content model category. So to summarize, we compared block

level and inline content types and which we know are not part of

the official HTML5 classification. But they're still used quite often in

literature and just in the regular coding and they're roughly equivalent to flow

content and phrasing content respectively. Next we're going to

talk about headings and we're going to explore some

new HTML5 semantic elements.