# ***POSITIONING ELEMENTS BY FLOATING***

[SOUND] In

this lecture we're going to talk about positioning elements by floating them.

Now knowing how to properly float elements is an essential skill to have, and

this is partially because most of the UIs today, web UIs today that are made,

are made by floating the elements.

In fact, by the end of this lecture, we're going to create a two column flexible

design where the columns will be flexible as we expand and contract the browser.

So let's jump into the code editor and see how this works.

Okay, so we are in sublime text and

we're looking at the file named floating before that HTML.

So let's go over the structure of this HTML document real quick.

We have our regular h1 here just to announce what that is, and

then we have a div with four paragraph tags and

a section tag with some text in it, and the paragraph tags ID p1, p2, p3, p4.

If you look at our div section, we styled the div tag,

its background to be a certain color.

Play video starting at :1:4 and follow transcript1:04

Basically a light bluish color, and every single paragraph tag was styled to

be 50 pixels wide and 50 pixels tall, and we gave it a nice one pixel border.

And just to differentiate between each paragraph box,

we gave it a different background color to each one.

So let's go ahead and take a look at what this looks like in the browser.

Play video starting at :1:26 and follow transcript1:26

Okay, so here are the four paragraph boxes that we had.

Okay, so let's go ahead and float the very first paragraph box to the right.

We'll say, float: right, and we'll go ahead and refresh the page, and you see

that the box jumped all the way to the right, top edge of the containing element.

Now, a couple of quick things to notice right away.

First, is that you could see that the rest of them moved up as if the maroon box

isn't there anymore.

And the reason this is happening is because when you float elements,

the browser takes them out of the regular document flow.

And this is why the second paragraph box moves into the original spot of the first

one, because as far as the regular document flow is concerned,

that spot became empty.

Second thing to notice is that even though each one of these paragraph boxes

has a margin around them, as you can see that there's a margin between

each one of them right here, that margin collapses as we've learned before

when it is touching the margin of another element.

However, when it comes to floated elements, the margins never collapse.

And this is why you now seeing the margin around the element on top.

Also, if you paid close attention, you will notice that the blue,

light blue box, the development became shorter.

To really illustrate what's going on, let's go ahead and remove the float right,

and instead make every single paragraph box float to the left.

And to ensure that the boxes aren't squished right against the other,

let's go ahead and give it a margin-right of let's say 10 px.

Let's go ahead and refresh the page and you could see that now they're all floated

to the left and there's a ten pixel margin between them.

However, what happened to our div?

Our div seemed to have completely collapsed and the only thing that's really

keeping it open at all is the text that we've put in here.

This is happening because of the same point that I made before.

When you float elements it takes them out of the regular document flow,

therefore it collapses and hide, and

surrounds the last section element which has actually some text in it.

In order to correct it, we need to tell the browser that when it

comes to this section element, the browser should resume the regular document flow.

And the way that we do that is by using the clear property.

So let's go ahead and go to our styling and define section target and

wel say, clear: left.

And what this is doing is telling the browser that the section element is

announcing that nothing should be allowed to be floating to the left of it.

And since every single paragraph that we've been floating is floating to

the left, when we save it and

refresh, this should jump right below here, and so it does.

And this is not restricted to the non floated elements.

We can ask a floated element to resume its regular document flow, and

then float it again.

So for example, the reason this box is situated right here is because there's

something already floated to the left of it, but

if I tell the browser to go ahead and resume the regular document flow on it,

it should jump to the next line and then resume floating from there on.

So that's box number three, so let's go ahead and go over here and say clear, and

since things are floated to the left of the box, we're going to say clear: left,

so nothing should be allowed to flow to the left of it.

We save and refresh, you can now see that

box number three moved in its own line again and is floated to the left again.

There's one more value of the clear property that I should explain before we

move on to try to create our two column layout, and that value is both.

So you could say clear both.

So let me show you a circumstance where that would actually apply.

We're going to reset our page back to the way it was, and we'll go ahead and

remove the floating elements here, and we'll refresh, so

this is how it was when we started.

So let's go ahead and float the first element to the left as we did before,

and we'll float the second box to the right.

Okay, so now if we refresh the browser you can see that both boxes floated, and

you can see the other two kind of moved into their places.

So we already know how to deal with that.

In order to move them to the next line, we can just say clear.

And now we need to clear what?

Now there's a box floating to the left and there's a box floating to the right, so

we can say clear: right.

And when we refresh the browser this will go on it's own line.

However, we can equally say clear:left because

there's a box that is on the left as well.

So it seems like in this particular circumstance,

it doesn't make any difference whether you say left or right.

However, it will make a difference if I make the right box much taller than

it is now.

So I'll go ahead and go to box number two, and

I'll make its height something much taller, let's say 300 pixels.

When I refresh the page, you could see that it really went all the way down and

it's totally ignoring the fact that I want to clear this element,

meaning I want it really on the next line by itself.

And the reason it's happening is because I'm actually clearing this element

only from the left floating elements,

but I'm not clearing it from the right floating elements.

So therefore, if I wanted to, I could say here, clear: right, again and

when I refresh, it will jump all the way down.

But the problem is, is that what happens when this element becomes too tall?

Well, then it will start overlapping again, right?

So if I make this element, let's make a little bit less, let's make it 100.

And when we refresh you can see now, it's still not overlapping

because we're clearing to the right but lets now make element number one polar,

lets make it's height say 150 pixels and refresh you can see now it's overlapping.

So, in order to ensure that both columns, or both left and

right Floated elements are not encroaching on our space here of the third element.

We could just say, clear both.

And refresh you could see that no matter which one is going to be taller

it will still insist that nothing should be floated to its left or

to its right, and therefore it will be on it's own line.

Okay, so let's go ahead and try to achieve that two column layout design.

And in order to do that we're going to take a look at a file called two column

before that html and it's located in the same folder, examples lecture 21.

An HTML structure is basically very similar to the file we just looked at.

The differences here is that I only have two paragraph tags inside that div,

I removed the other two, and we still have a a section tag inside of it.

And what we want to achieve is that this first paragraph should be left column and

the second paragraph should be the right column.

So let's see what we did here.

So first of all, for now, we're keeping the background colors of all the elements.

And on the paragraph tag we said the width, and the width we did something

different this time, we didn't specify pixels, we specified percent.

So we're saying the width should be 50% of its containing element, and

the containing element is the div.

Well, the div is a block level element, so it tries to fill in the entire width of

its containing element, which is the body or the entire browser.

So basically what we're saying is that a paragraph should take 50% of the entire

view port.

We'll also give it a one pixel border and we floated each one of them to the left.

So let's go ahead and take a look at the browser to see what this turned out to be.

And looking at the browser, not quite exactly what we wanted.

The second paragraph is still on the bottom.

Why is this happening?

Well, the reason this is happening is because we're still using the default box

sizing, which is content box.

Then we're saying that we want the box to take 50% of the screen and

then we're adding that one pixel border, that really breaks the whole thing.

So if we remove that pixel right here, and

refresh the browser, now it's floating exactly how we want it.

And by the way, this is a great illustration of what happens to floated

elements when they can't fit on the same line.

And what happens is they float to the next line.

And this is actually what happened when we had the border pushing the size

of the box a little bit too much that it didn't fit on the same line anymore.

However for now we do want the border, and

if we get the border then we can't have the floating, so how do we fix this?

Well hopefully you still remember how to reset the box-sizing, and

the box-sizing we're going to say is border-box.

And once we do that, the border will be included in the 50% of the width, so

now we save the file, we go ahead and refresh, and

now we see even with the border we can have both of them floating side by side.

Let's enhance out layout a little bit by giving some padding

to the content inside each paragraph.

So let's say padding of, let's give it 10 px all around.

Let's refresh and we can see now that we have the content separated from its

border a little bit better.

And finally we can actually remove all of these background colors and

the border because they were only there to help us visualize things, so

we're going to go ahead and comment them out and

refresh, and now we see we have a two column layout.

Let's make this a little bit bigger, and as you can see now,

the layouts have a two column layout, and as I squeeze the browser,

the two column layout stays, and not only that, they're flexible.

And the reason they're flexible is because we didn't specify them with a particular

pixel size, we specified them as a percentage of its container element.

So in summary, floating elements can produce very flexible layouts.

Floats are taken out of their normal document flow, and

they're positioned at the top left or right edge of the containing element.

Play video starting at :10:22 and follow transcript10:22

Also, floats don't have vertical margin collapse, so

previously hidden or collapsed margins will show up once you float the element.

And remember that to resume normal document flow, use the clear property.

Either clear left, clear right or clear both.

Next, we're going to talk about relative and absolute element positioning.