





[SOUND]

Pseudo-class selectors address targeting only the

structures that can be targeted by simple combinations of regular selectors, or targeting the ability to style based

on user interaction with the page. For example, we would want the styling of

an element to change if the user hovers or moves their mouse over that element. The way you specify pseudo-class selector,

is by specifying some selector that we all ready know about with a colon

and a predefined pseudo-class name. Now, there are many pseudo-class

selectors that exist. In this lecture we're going

to cover five of them. The first four link, visited, hover, and active are pretty basic and

is used very, very often. The last pseudo-class will cover nth-child

is a fairly powerful pseudo-class and can get pretty complicated, but its basic uses are fairly

straight forward as we will see. So let's jump in straight into

the code and see these selectors work. Okay, so in Sublime Text I'm looking at

the file called pseudo selectors before that HTML that is located in

the examples lecture 15 folder. Let me show you what this HTML

page looks like in the browser. So this page has a header, and in the header we have an unordered

list which has a bunch of links. As you can see it's showing up as

a regular list with bulleted points. Followed by a section which

has a whole bunch of DIVs and each one is named DIV 1, DIV 2,

DIV 3, all the way to DIV 20. And the first thing I'd like to do is

I'd like to style this unordered list as a menu. It's a pretty common practice to take

an unordered list and style it as a menu. So let's go ahead and

take this unordered list and style it as a bunch of menu buttons. The first thing we want to do

is target these list items. So the way we're going to do this is by

using our descendant selector by saying header li, and we'll turn off the bullet

points by saying list-style: none. And if we refresh the page we now see

that the bullet points went away. The next task is to style the links that

are sitting inside the li elements and to make them look like buttons. However styling links is not exactly as

straight forward as styling a regular element, and

that's because links have states. And these states can be expressed

using our pseudo-classes. So let's go ahead and do that. So here we're targeting two states. One is a's being a link that it is. And another one a visited. And this is very often the practice

that you group together these two. And what basically that means is well

link is at the way it is link, and visited means that HTML allows that

after you click a particular link that a different style can be applied to

that link than an unclicked link. In our case, however, we don't want

to differentiate between the two, so we'll style them both together. Okay, so

let's start by removing the underline that is the default styling for

every link. Once we save that and refresh,

we see that underline is gone. Okay, so let's add some other styles. We'll set the background color to green. Border, one pixel with solid,

meaning solid versus dash or something like that, and blue. And we'll set the color of the text

inside these links to black. Let's go ahead and refresh the page, and

here we'll see we have a blue border, it's a green background,

and the color is black. Now I spoke to you before about

the fact that the a element is both sort of an inline element and

a block level element. However, that's only as

far as inclusion of other elements inside of it is concerned. As far as this default display, it's

actually displayed as an inline element. Well we don't want that for our buttons,

we want them to block level element. Well I can change that very quickly

by just saying display: block. And once I say display: block and refresh you can see that the block

level elements do what they always do, try to fill up all the space as

they possibly can within that line. So I'm going to have to

constrain it a little bit. Let's go ahead and

give it a width of 200 pixels. We don't want our button to be that wide,

and we want our line and menu items to be center. And we'll also separate the different

menu buttons from each other by giving it a margin bottom of one pixel. So let's go ahead and refresh that. And now we see we have three

little nice buttons in our menu. However we're not done because

what we would like to do is on the user hovering over these buttons,

besides changing the cursor to a pointer, I'd like to have some sort of

interaction in the style and kind of show the user that they're

hovering over these buttons. So in order to do that I have to

define two more states of the link. One is a hover and the other a active. Now hover is exactly what it sounds like. So when the user hovers over the element,

in this case the a element. An active is that state when the user

actually clicks on the element but hasn't yet released his click. So he pressed the mouse button down but

hasn't yet released the button. In our case we don't want to differentiate

between the hover and active, so we'll keep them grouped together. Let's set the background color to red and

the color of the text to purple. We'll save it and

preview it in our browser. So now when I roll over, you can see that

the color of the background changes to red and the color of the font is changing to purple even though combination with

red it's kind of actually looking blue. Now these buttons still

retain their link behavior. So if I click on this link, I will go

to the Facebook page that happens to be the fan page that I created

specifically for the courses that I'm teaching on Coursera, which is this course

and the next one, which is an angular.js. As a side note,

please go to this page and like it. You'll be able to follow

the latest updates and I will be posting some optional material that might be interesting to you

that's related to web development. And please sign up for the mailing list

that I'll be able to notify you when some new, interesting material comes out. Obviously all this is completely optional. In it's simplest form, the nth

child pseudo-selector allows you to target a particular element within a list. For example I'd like to target

this Facebook fan page link and I'd like to make it bigger just so you

remember to click on it and like the page. So let's take a look at

the HTML where it's located. It's located in the header section and

there is a URL and there are three li elements in here. The Facebook link is

the third one in the list. So I could go over to my style and I can say that I want to target the li

element that is inside the header element. And I want it to be nth-child(3),

so that's the third link. And now I can target it and say for example, font-size to be 24 pixels,

so I'll make it a little larger. So if I refresh the page,

you'll see that only the third link, the third menu button, became bigger. The rest of them stayed exactly the same. But nth child pseudo selector can

do much more fun stuff than that. For example we have a list of

DIVs here that looks pretty dull. So what I'd like to do is I'd like to make

it look like a list and what I want is that every odd member of this list should

get its background color to be gray. So the way I could achieve that is by

targeting the div inside the section and I'll specify the nth child again. And instead of specifying the numbers one,

three, and so on, I could specify odd. And if I do that and say background

color gray and refresh my page, all of a sudden I have a list that's looking

much, much easier to read than before. And by the way, just like regular

selectors can be combined, pseudo-selectors can be combined as well. For example, if I would like to have

the fourth element in this list have a different hover behavior, since

right now nothing really changes when I hover over the fourth element on the list,

I'd like to have that changed. So let's go ahead and target again,

section div, and target the nth child. This time I want the fourth child to be

targeted, but I want to specify that its hover behavior should be different, so I'm

going to target the hover behavior part. And I'm going to change this background

color to green and its cursor to pointer. So let's go ahead and refresh the page. And now when I hover over number four,

the cursor turns into the pointer and the background of that

element turns green. I'm sure by now you see how

powerful selectors can be, especially using

the pseudo-class selectors. One quick word of warning though,

make sure your selector is still readable. These things can get very

complicated very quickly. And if you make it such that other

developers, and you in a month or so can't really understand fairly

easily what you're targeting, this can become kind of

a maintenance nightmare. So simple and readable is definitely

preferred to complicated and tricky. You might be able to

impress people with it, but it's certainly not good coding practice

to make this super complicated. Next, we're going to talk about

where do we place the CSS styles. What is the best place to place it and what the differences

between those places are.