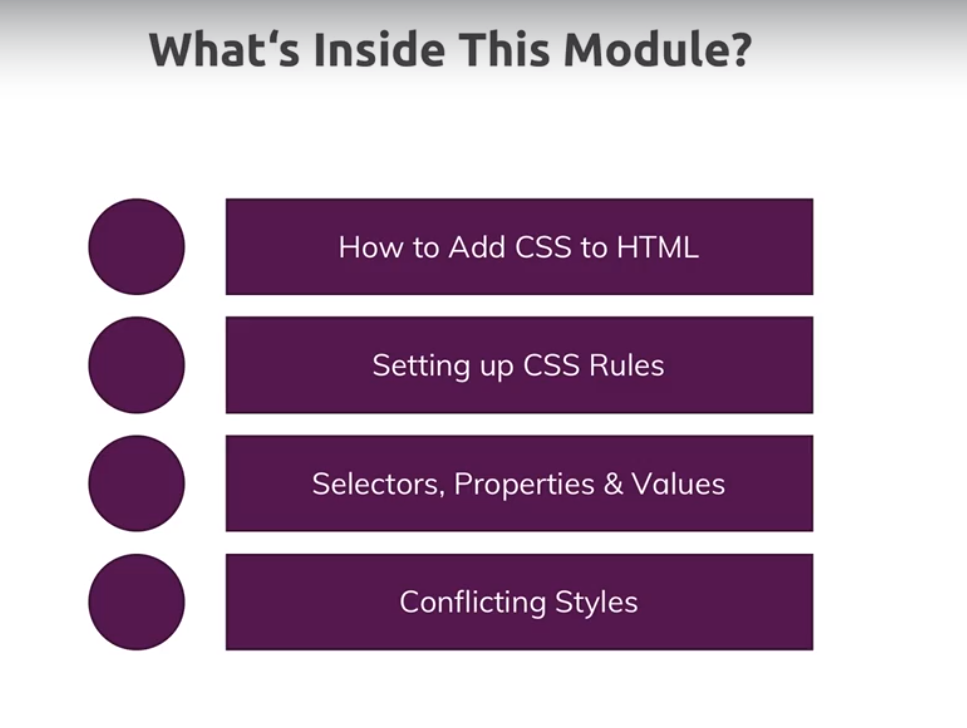
11)Module introduction



How to include css file, do this-

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>uHost</title>

<link rel="shortcut icon" href="favicon.png">

<link rel="stylesheet" href="main.css">

</head>

15)Appying additional fonts and importing Google fonts

Here we saw that we can use font-family property to change the font style. As value we can assign default font like sans-seriff, this is special keyword which will use the default set up in your browser.

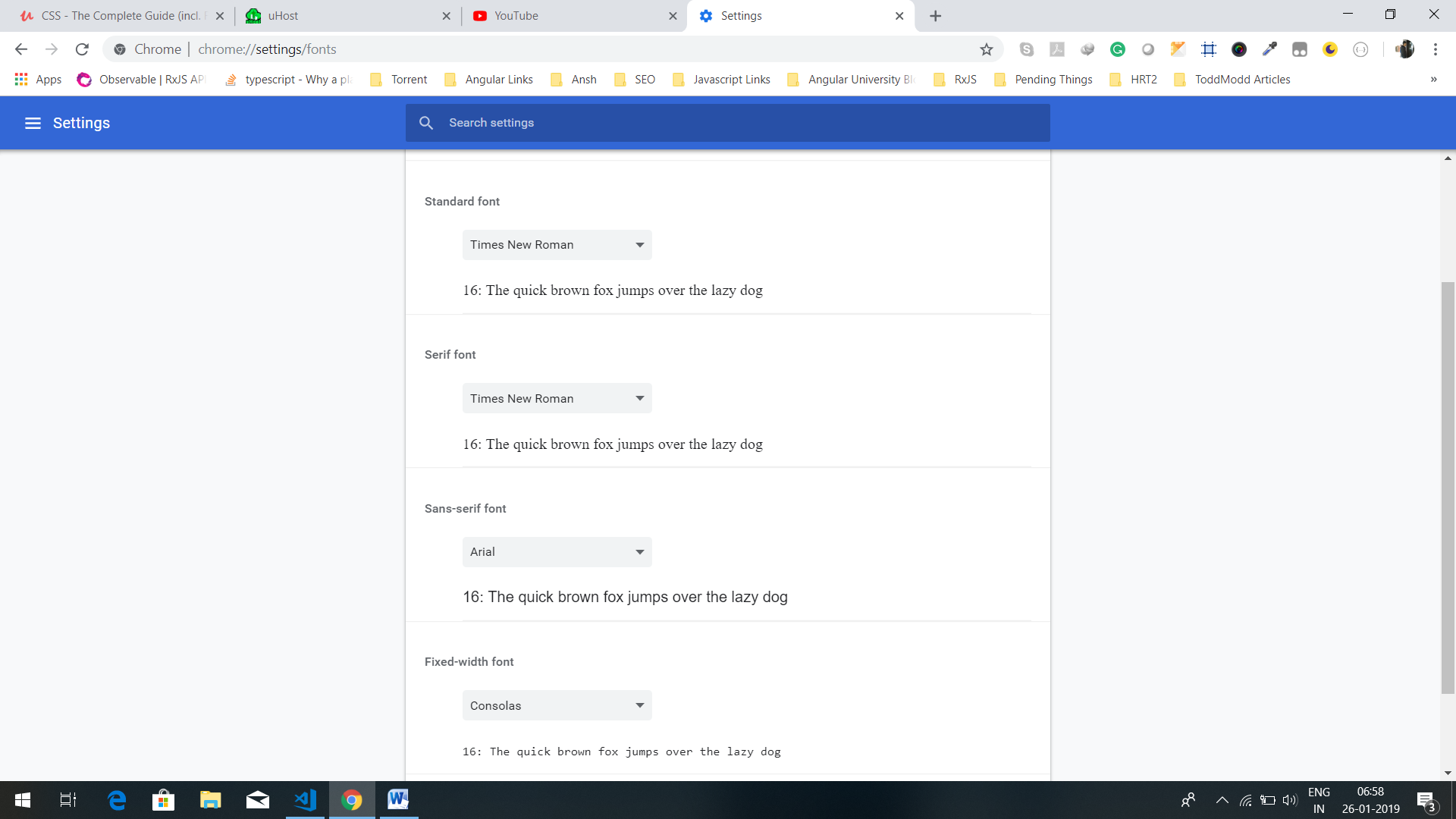
h1{

color: white;

font-family: sans-serif;

}

Now if we see font style sin our page is changed, from where is this coming from. In chrome, go to settings ->fontsettings, you will see this screen-



This is from where these fonts are coming from. we can change them how they look. every browser has such setting. This standard font is used if do not specify anything i.e if we do not use font-family property.fixed-width, last one is used for code snippets.

Now you are not limited to use font-families included on your system. However serif,san-serif and mono space are great values because they will use browser defaults. So you can rely on these keywords picking a font that looks at least to some extent in way you want. sometimes you want a specific font though and you can’t rely on that font being installed on the machine. So use the font which is not necessarily installed you can use tool called google fonts.

Google it and and open their ebsite. There you can see many fonts.. you can serch fonts. If you want to include hat font, then click on + sign. You wil see link that you need to add to head section of your html and then you can see value for font-family property. Make sure import of font is added before import of our css file so that we can use it our css file.

Head section-

<head>

<meta charset="UTF-8">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>uHost</title>

<link rel="shortcut icon" href="favicon.png">

<link href="https://fonts.googleapis.com/css?family=Anton" rel="stylesheet">

<link rel="stylesheet" href="main.css">

</head>

In css-

h1 {

color: white;

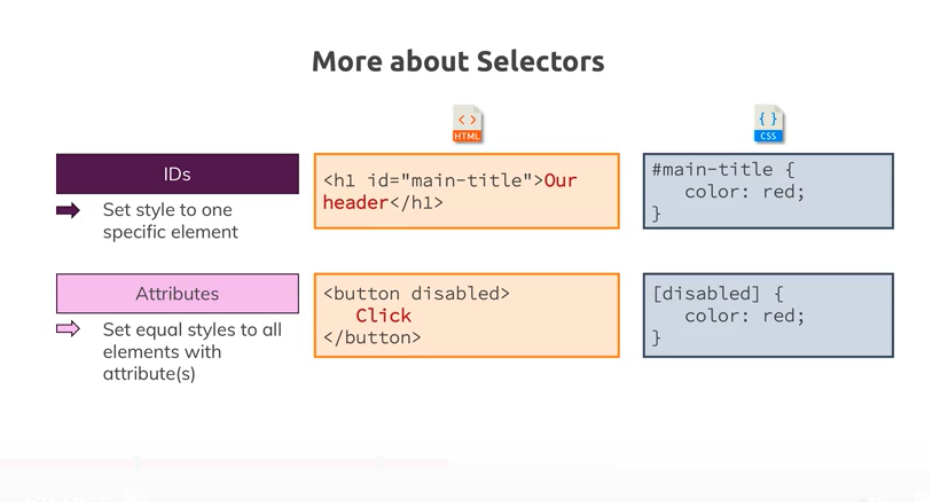
font-family: 'Anton', sans-serif;

}

16)Theory Time Selectors



Here universal selector is new for us. We rarely use this one. But in this course we will see 1 case, where we will want to use it.



17)understanding the “cascading” style & specificity

Lets say we have this html-

<h1 class="section-title">Choose your plan</h1>

In css we have-

.section-title {

color: #2ddf5c;

}

h1 {

color: white;

font-family: 'Anton', sans-serif;

}

h1 {

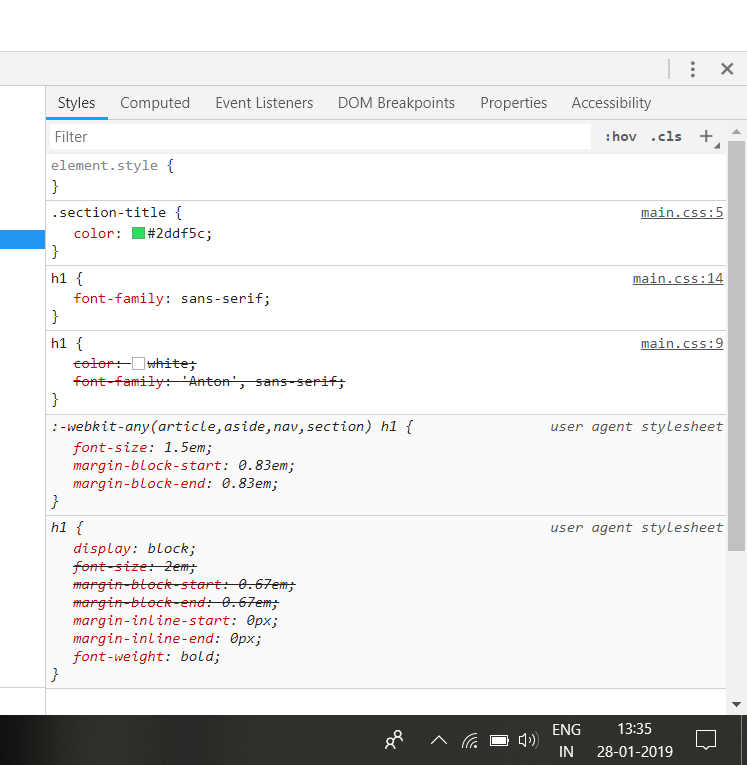
font-family: sans-serif;

}

Now element is selected by all selected.

Here we have h1 selector 2 times, if they have same styles, the same selector which is defined later will overwrite first one. But class styles are more specific so it will always overwrite element selector. so order of defining only matters if selectors have same specificity.

You can check in chrome dev tools, inspect a element. So there style are aligned in highest to lowest priority. Highest priority is to inline styles, first style in chrome dev tools which is empty, is inline style.



So inline style has highest prority. Then we can see that class style has high priority than tag selector. then we can see that when selectors have same priority ,than order in which they are defined matters. Here we can see h1 which has defined later overwrites the style(font-family) defined by previous h1(it comes before first h1 in chrome dev tools). so you can say that second h1 selector has highest priority than first one.

Also we have browser defaults, they have low priority, but they are there. We can override them but we will come to that later.

The fact that we have multiple rules affect the same elements is cascading part of the name.css stand for cascading style sheets and cascading simply means multiple styles or multiple rules can be applied to same elements. Now these rules may lead to conflict though, like we define color property in both class selector and tag selector. that is conflict. Now to resolve these conflicts , css knows a concept called specificity. there are clear rules defined in css specificity that define how these conflicts should be resolved and which type of selector has higher specificity. Here are overviews. Here is the order-



Note- universal selector has lowest priority.

We have’t looked at pseudo-element an pseudo-class selector. highest priority is to inline styles, we should’nt use them. But if we add them they override all other styles.

Now there are more rules conneted to specificity, some advanced things connected to thing like inheritance which we have’nt looked at yet. But these rules here are important to keep in mind.

18)understanding the inheritance

Inheritance means that an element also inherits some styles of parent element. We are not using inheritance , let use it in our project. Lets add it by setting the global default font of project. One approach to set global default font will be to use global selector-

\* {

font-family: 'anton'

}

Now each element will have this default font, but if we give font-family or a selector, it will override this \* selector because \* has lowest priority. So this is what we want to do.

But problem with \* selector is that it is very inefficient, the way css works, css has now, have to parse all our elements on the screen. So we will not use \* for setting default font-family.

For this purpose we style body tag, it wraps all other content on page. Lets do this-

body {

font-family: 'Montserrat', sans-serif;

}

#product-overview {

background: #ff1b68;

}

.section-title {

color: #2ddf5c;

}

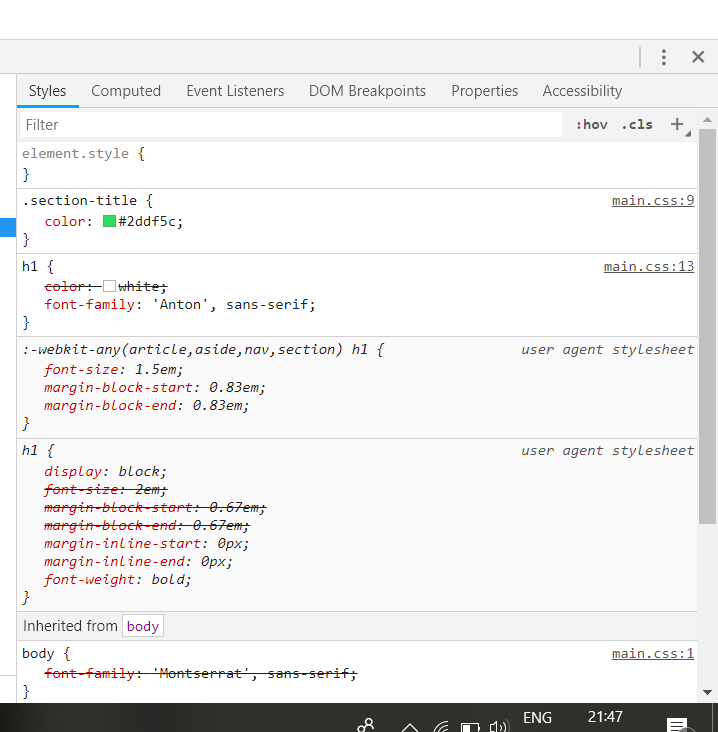
h1 {

color: white;

font-family: 'Anton', sans-serif;

}

Now if we inspect h1 element, we will see-



At the bottom we can see inherited from body secton. Note that this is even down the browser defaults.

So elements inherit styles from their parents, direct or indirect parent and not from just body. Body is no direct parent of h1, but it is super parent. Therefore some styles not all , there are some exceptions(we will dive into them later). So some styles,especially font related styles are passed down to childs. However inheritance has very low specificity. It comes even below universal selector. so it means where we select element directly, (it can be tag ,class,id,browser default) will always override inherited style.

So inheritance is important concept for passing styles down without explicitly selecting an element and especially for things like font sizes, font families, this is extremely useful bcoz you typically want to have one and the same style for majority of text on your screen. So setting it up in body section is great way as it will make sure that you can use inheritance.

Important- any direct selector has higher specificity than inheritance.

19)Adding combinators

Html-

<main>

<section id="product-overview">

<h1>Get the freedom you deserve.</h1>

</section>

<section id="plans">

<h1 class="section-title">Choose your plan</h1>

<p>make sure you get the most out of your money</p>

</section>

</main>

Css-

body {

font-family: 'Montserrat', sans-serif;

}

#product-overview {

background: #ff1b68;

}

.section-title {

color: #2ddf5c;

}

h1 {

color: white;

font-family: 'Anton', sans-serif;

}

Now what we want to is we do not want second h1 tag to have styles as first one(one with class section-title). We overwrite the color in class selectors but font is same in both. we want to have default font in second h1. One way to do that will be-

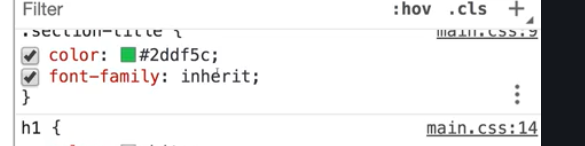
.section-title {

color: #2ddf5c;

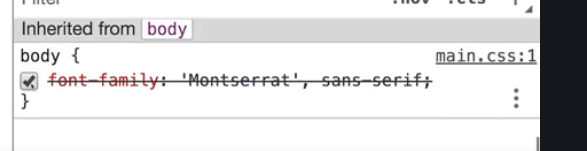
font-family: inherit;

}

If we go to chrome dev tools, we can see that-



If we scroll down-



Here it seems like this style is not applied but it is applied because we are using inherit value.

Here in class selector, we say that we want font-family that in is inherited. Here we will get font-family that we have defined in body tag. Here we basically increase the specificity of inherited style. but this is not best way, because if we have more h1 and we want it to have same font-family, then we have to add font-family:inherit to all of them. Instead we can define where we want anton font-family, then rest of h1 will automatically use inherited font-family.we dnt s=have to set inherit value.

One way can be we can place a class on first h1, then we can define font-family there. One other thing that we can use is combinators. Combinators allows us to combine multiple selectors and be more precise , what we want to select. So we do this now-

body {

font-family: 'Montserrat', sans-serif;

}

#product-overview {

background: #ff1b68;

}

.section-title {

color: #2ddf5c;

}

#product-overview h1 {

color: white;

font-family: 'Anton', sans-serif;

}

We say that all h1 inside product-overview id should have this style. Now h1 does not need to be direct child of product-overview. It can be child at any level.so it will also work in this case-

<section id="product-overview">

<div class="sumit">

<h1>Get the freedom you deserve.</h1>

</div>

</section>

It means this style will be applied to all h1 which are nested somewhere inside the element which has id, product-overview.

Advantage of this as compared to first approach is that, now we can have any number of h1 inside product-overview and all will have same styles. we do not need to add classes manually to each h1 . so what we are using here is called combinator, because we combine different selectors, as a side note, **if we use combinators, we also increase the specivity** . so if we have this-

#product-overview h1 {

color: white;

font-family: 'Anton', sans-serif;

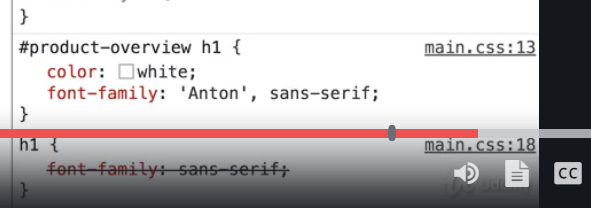
}

h1 {

font-family: some other style;

}

Chrome dev tools-

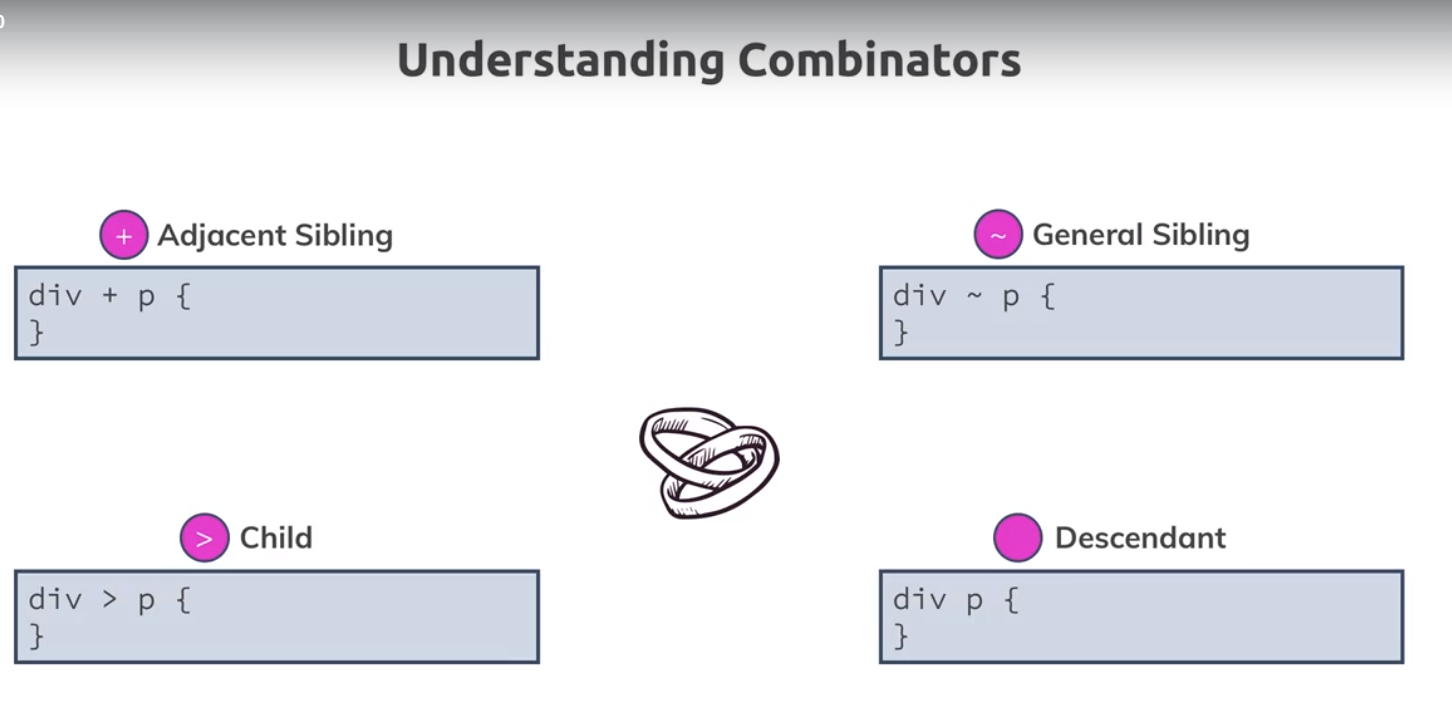


To all h1’s inside product-overview id, first style will be applied. it is because combinatory has more specifivity. It is clear by order of selectors in chrome dev tools.That is last important piece about specivity, the rule with more information to it , so as to say( we pass more to it , as we want to narrow down to which h1 we want to provide this style) wins over rule with less information(here only h1 selectors is rule with less information).

20)Theory Time- Combinators

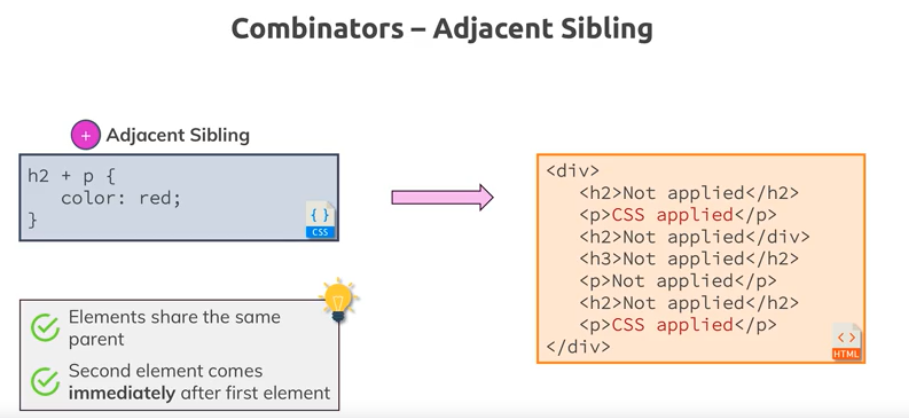
In last lecture we had a look at one combinatory. Now there are more combinators, four very important to be precise. Now combinators allow us to be more clear about our rules and select elements by passing more information to the selector. now oyu can combine multiple selector not just two as a side note, and as I mentioned you can combine them with four important type of combinators.

These are-



First one is one is added by adding + between selectors. and again There can be more than 2 selectors, you can something like his- div +p+a.

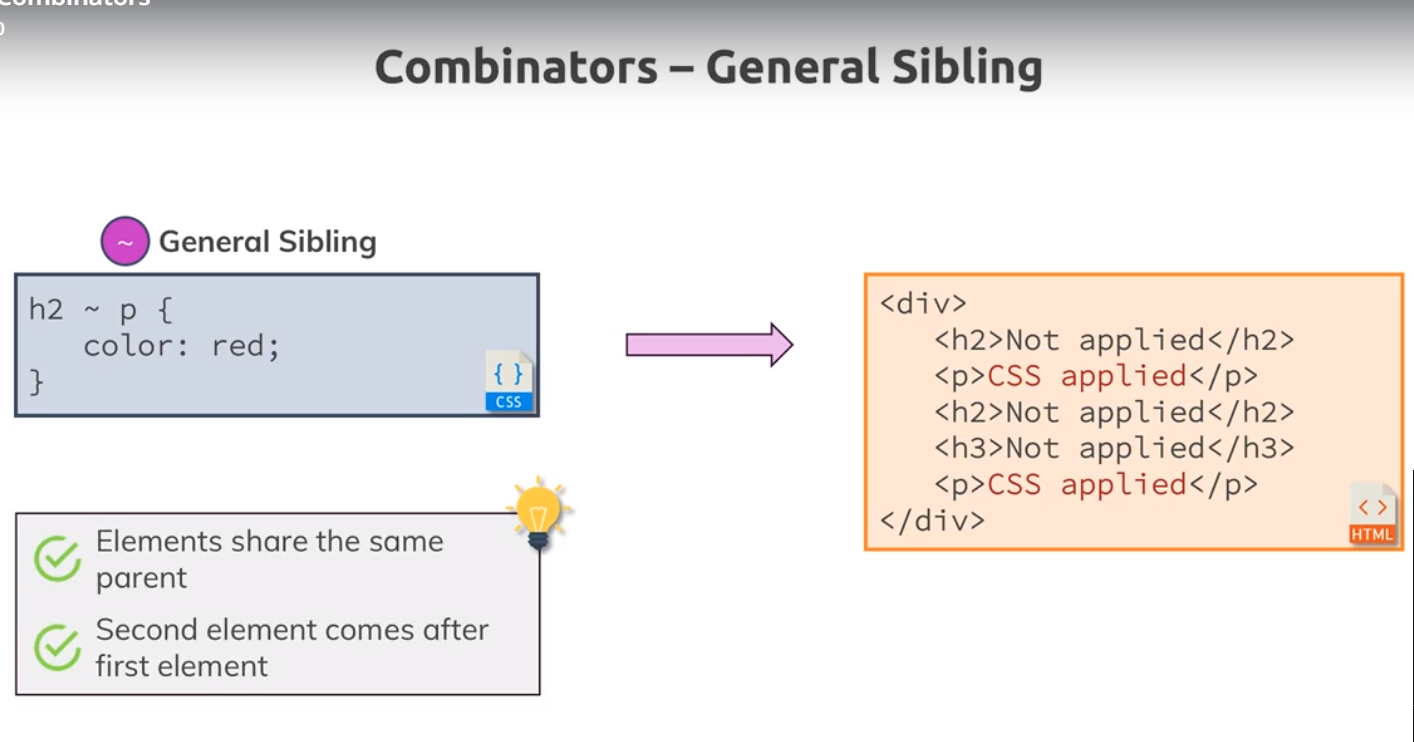
Now lets see each one of them in detail-



This selectors selects all p tags are immediate after h2 tag. This selector does not select the second p tag because it is not direct sibling of h2 tag. Another important thing to keep in mind is that both h2 and p tag need to have same parent.

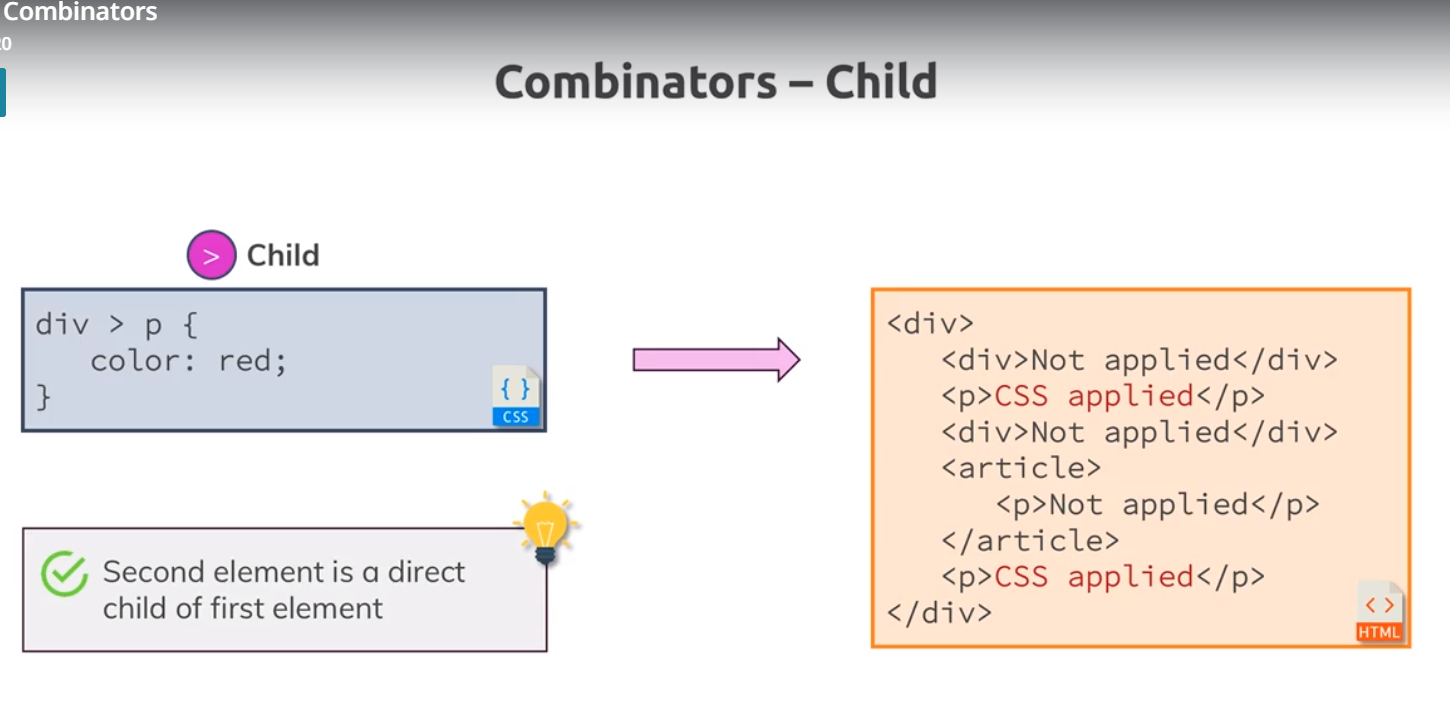
Also we can combine more than 2 selectors, like this div +p +a

Lets look at general sibling now. Its connected to first one but is more flexible.

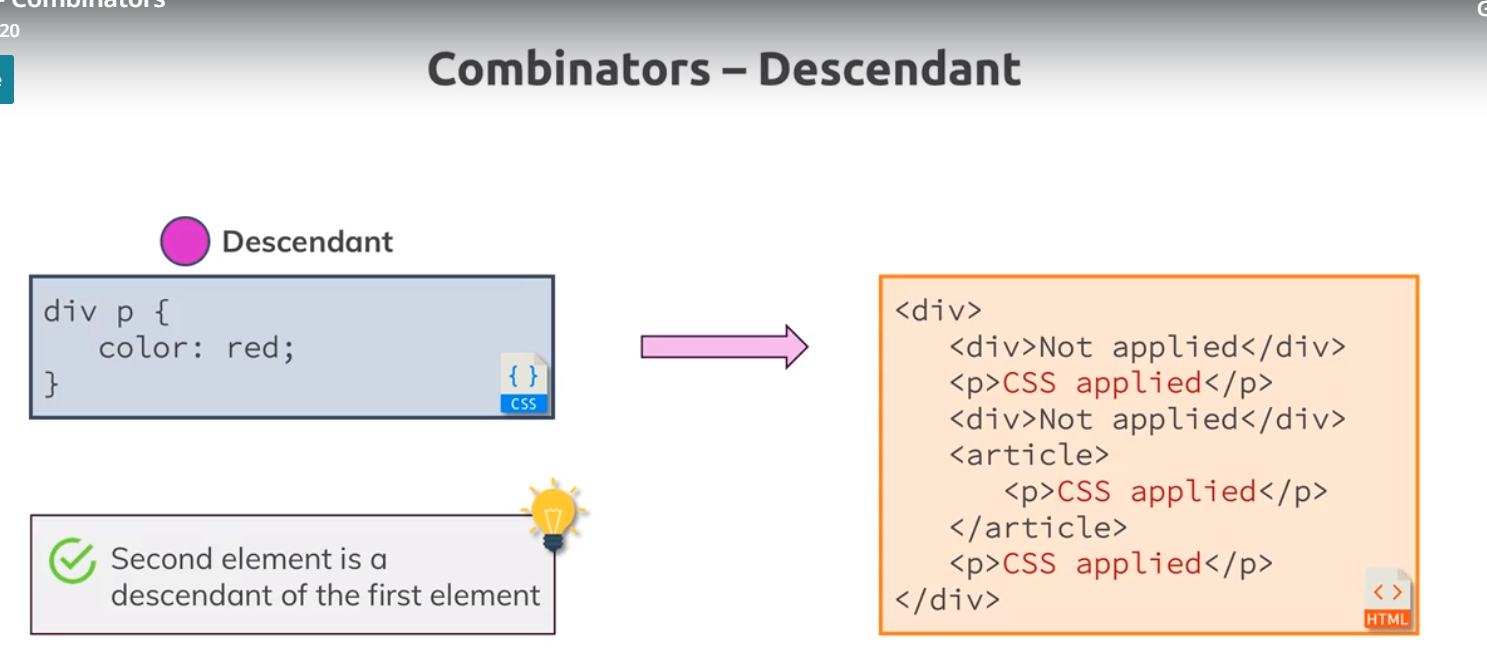


Here last p tag also gets styling even through it does’nt directly follow the h2. Fir general sibling only thing required is there needs to be h2 siblings, it does’nt have to be directl sibling of h2. Important things is, p tags needs to come after h2, but it doesn’nt need to come directly after h2.

Child combinator selects the direct child. P tag which is nested inside article tag is not selected. As it is not direct child of div.



Note if had 2 p tags in div, then this selector will also select them both.

Last one is descendent. Here level does’nt matter. Second tag can be nested inside first one, at any level. Here second p tag is also selected, because its super parent is div. it doesn’t have to be direct child, it only needs to be descendent.

This is combinatory that you use most often.

Definitely use combinators , if they allow you to be more precise but you should be aware that direct selectors with out combinators are showing a little bit of better performance. That being said , its not like combinators are super bad and it also matters on what you want to combine. For ex, this combinatory-

#product-overview h1 {

color: white;

font-family: 'Anton', sans-serif;

}

It has very good performance. Because we are very clear about element(h1). it would be worst if we have \* like this-

#product-overview \* {

color: white;

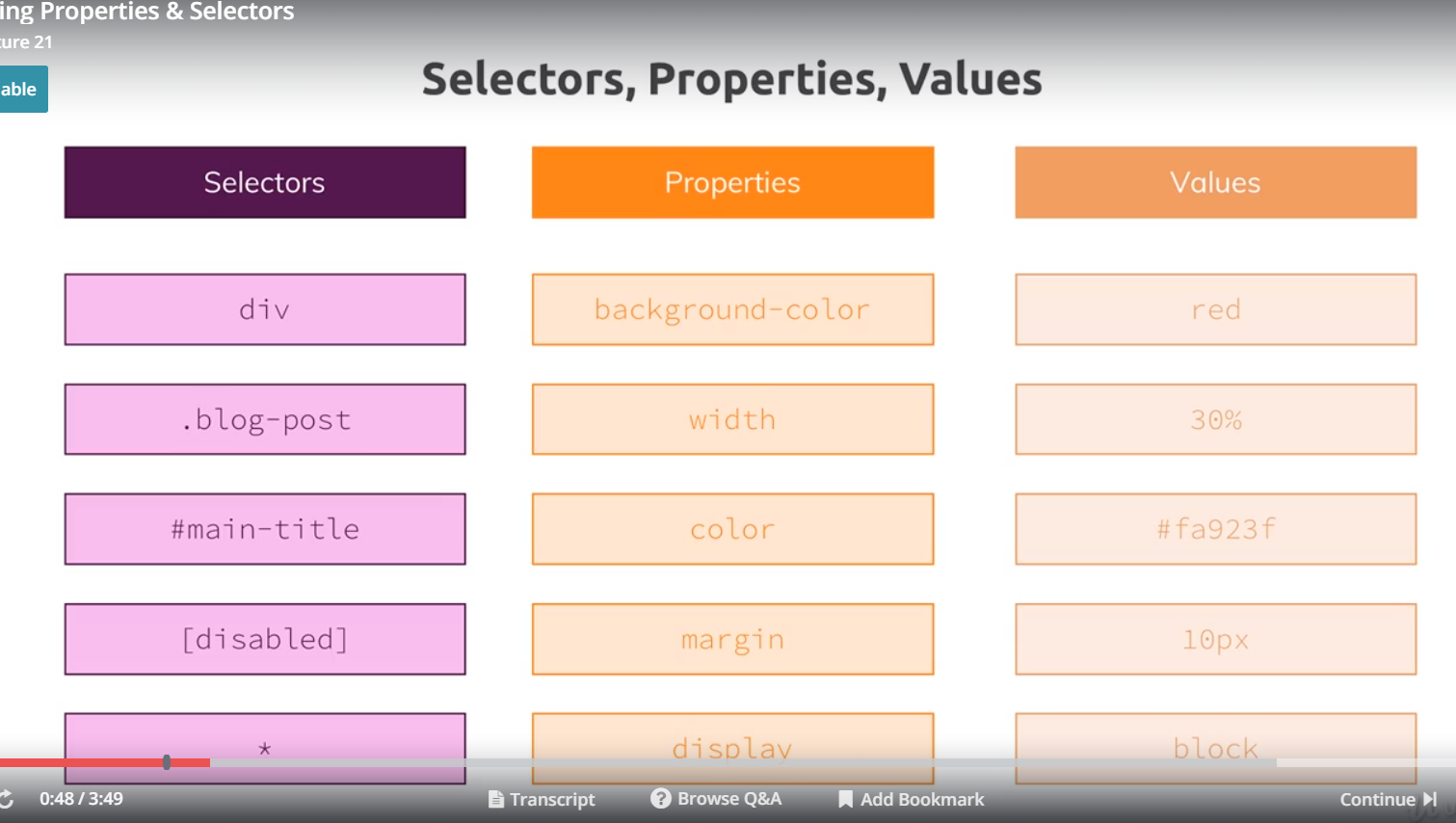
font-family: 'Anton', sans-serif;

}

This has very good performance because we are using h1 and second reason is we are using id, which has awesome performance because there are’nt many id’s on our page. By the way performance for classes is also pretty great. So classes, id ‘s have pretty great performance, combinators which use them also tend to do pretty good.

So combinators can be really helpful, last combinators is probably the combinatory which you will use most often and they allow you to really ensure that you only style the element in parts of your app where you want to style them.

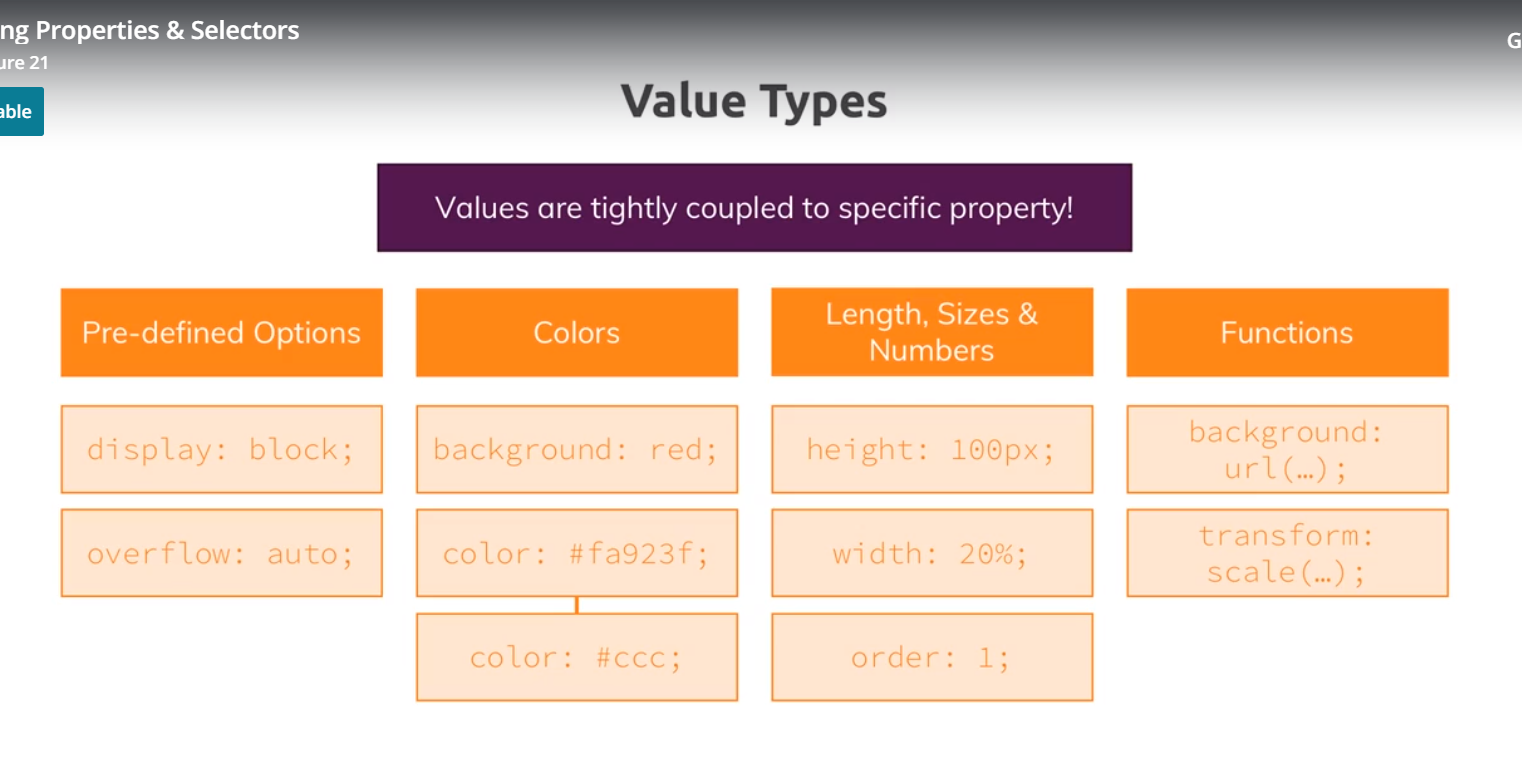
21)summarizing properties and selectors



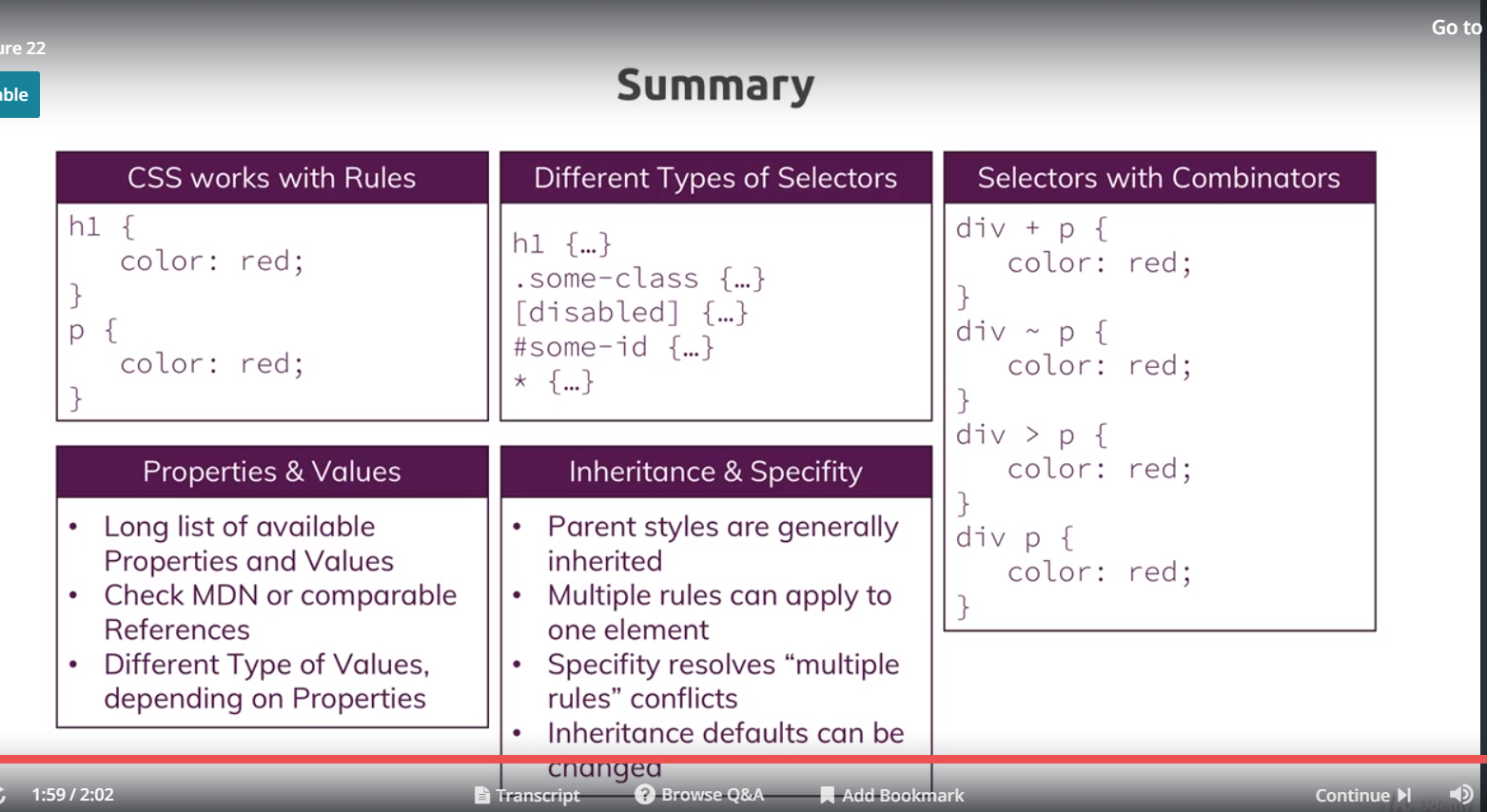
If you wonder, how many css proeprties and values exists.let me start with proeprties. Visit this link-

<https://developer.mozilla.org/en-US/docs/Web/CSS/Reference> (added to bokmark, folder, css Links)

here go to **keyword index.** You will not only see all properties but also pseudo classes and elements(something which we will cover later). This is place for looking for property that you need or finding something more about property that you found in solution on stack overflow. But you dnt need to memorize them all, there are only hardly 20 properties with which we work on day to day basis. I just pointed you there so that you need where to look things up. Now coming to values, you can click on property to see what kind of values it accepts. I have divided values into 4 categoeis, there can be some exceptions. But generally there are 4 tyopes. These are-



22)Summary



23)Useful resources and links

* Complete MDN CSS Reference (don't learn this by heart!): <https://developer.mozilla.org/en-US/docs/Web/CSS/Reference>
* Are you a reading guy? Find written CSS docs on MDN: <https://developer.mozilla.org/en-US/docs/Web/CSS>
* Common CSS Properties Reference: <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Properties_Reference>
* CSS Combinators: <https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/Combinators_and_multiple_selectors>
* More details on CSS Specifity: <https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity>