This lesson is going to be about **hierarchy stylesheets**. In this section, we're going to talk about the foundations, namely, CSS resets and normalizers. We're going to build our hierarchy on top of resets and normalizers. And in the next section, we're gonna examine how to establish hierarchical styles from text styling to component styling to some specific overrides.

This section is going to be about **CSS resets and normalizers**. We're going to talk about the different roles that resets and normalizers fulfil and we're going to compare and contrast the two approaches. We will discuss the advantages and disadvantages of using resets and normalizers.

We talked about the ITCSS architecture in the last section. Resets and normalizers are placed in the third layer of the ITCSS architecture from the top and this layer is called the Generic layer. The layers above the Generic layer have not much to do with generating CSS. In general, they might just modify the generated CSS.

From the Generic layers on first, we create and write actual CSS code. Elements, Objects, Components, and Trumps are going to contain the essence of our stylesheets. And all these stylesheets are built on top of a generic reset or normalizer. Let's compare resets and normalizers.

The main difference is that the CSS reset removes all built in browser styling. This means that every single element is going to look the same regardless of whether we are talking about an `h1` or a `div`, they are gonna have the exact same font size, exact same margin, padding and everything else. After a CSS reset is applied, we have to start styling everything from scratch.

There are obviously some CSS resets that remove just a fraction of all the built-in styles attached to some elements, for instance, just the margin and just the padding, and this way, an `h1` is gonna stay larger than an `h6`. Depending on whether you use a partial or a full reset, you're gonna have different challenges and the other layers where you write your actual stylesheets.

The role of the normalizer is that it makes elements appear consistent across all browsers that are supported. After applying a normalizer we can make sure that, for instance, exactly the same margin and exactly the same padding is applied on the same tag.

For the sake of consistency we will have to deal with less browser specific support in our stylesheets, which means that we can focus on what really matters, styling the element in a generic way for all browsers.

As I mentioned before using a CSS Reset, all elements look exactly the same. Obviously this is not going to be true in case of a minimalistic reset there some elements may differ because if you don't reset every single aspect of the element they are still going to look different.

In case of a normalizer, a heading will still look like a heading and `div` will still look like generic text. Normalizing stylesheets will not take away the styling of the elements. The style such is going to be normalized. Given that a full reset removes all styles, resetting the CSS achieves a consistent zero style baseline.

Normalizing just makes styles consistent, but the baseline is not going to be a zero baseline. It is going to be a non-zero baseline. This is because we keep the styles of the text. After performing a full reset, you have to decorate each and every element that you use in your markup, completely from scratch.

On the other hand, with a normalizer, you're free to take the basic style of elements, or you can also choose to decorate them further. If you decorate an element further, you have to consider the base style. A SitePoint article suggested a minimalistic CSS reset, namely, resetting all paddings and all margins for all elements.

So we can just use the universal selector to select every single element and apply

```
css * {padding: 0; margin: 0}
```

This is not a full reset because for example the headings are still gonna appear large, they're just not going to have any padding at any margin. This is very important because some browsers have different padding and margin settings than others.

And in order to make our styling group consistent in all browsers, we have to achieve a consistent 0 padding and 0 margin baseline everywhere.

We can conclude that the advantage of using a CSS reset compared to a nomalizer is that we're not going to get any surprises. We take full control of the styling and we have to build a styling from the ground up.

This is obviously more work than in case of a normalizer. This is one disadvantage of the CSS reset. In case of a normalizer, unless the normalizer is really bad, you're expected to write less code. The default styles are going to appear more intuitive because we are not going to wonder why a heading appears as if it was a `div`.

A CSS normalizer has some built in features that are going to be very useful for us. And they are also going to save some code for us. Performance-wise, even though browsers are very efficient, if our markup is large, typically a normalizer is faster than a reset. This is because some aspects of the styling are fully consistent and they don't have to be addressed by the normalizer.

On the other hand, other styling aspects are not going to be consistent. And they are just normalized by the normalizer for the browsers that have to be supported. I encourage you to visit these links so that you can get a feel for what a normalizer and what a popular reset looks like.

One of the post popular CSS resets I've encountered has been Eric Meyers CSS reset 2.0. You can even install it using NPM by typing `npm install reset-css`. Normalize.css is also a very popular addition to our stylesheets.

There are obvious is some other resets and normalizers available. For instance there is the popular HTML5 reset as well. Your task is to choose the reset or normalizer that you are comfortable with. And then, your style hierarchy is going to be on solid grounds. Sometimes, we don't even have to choose a reset or normalizer because there are frameworks that you're already using, and they include a reset or normalizer.