New and Upcoming CSS: Revisiting Cascade Layers

CSS layers resolve another problem with CSS specificity and rule order.

as I documented in: Looking forward: CSS Layers and the @layer at-rule CSS Layers or Cascade Layers are a way to group styles together and then apply them in a specific order.

All modern browsers now support cascade layers.

The idea behind layers is that you create layers as groups so that all the styles in a layer have the same priority and cascade together.

There are three ways to create layers:

1. Using the @layer block at-rule, with styles assigned immediately to it:

```
@layer base {
  &hellip;
}
```

2. Using the @layer statement at-rule, without any styles assigned:

```
@layer base;
```

With this system we can also define multiple layers and append content to each later.

```
@layer
  normalize,
  base,
  theme,
  content;
```

3. Using @import with the layer keyword or layer() function:

```
@import url(base.css) layer(base);
```

For simplicity's sake we'll use the firt method in the rest of this post.

The following stylesheet will create five layers for different groups of CSS. Rules in each group will cascade together and share the same priority within the author styles.

```
@layer normalize { &hellip; }
@layer base { &hellip; }
@layer theme { &hellip; }
@layer content { &hellip; }
```

Declarations in the layers defined later in the document will will against declarations in earlier layers. The specificity of the rules inside the layers is not as important as the order in which we declare the layers.

If there are declarations in the `theme` layer that match declarations in either base or normalize then theme will win since it's declared later in the document but it won't win against declarations in the `content` layer since content is declared later in the document.

As Bramus points out in his article about layers:

> Once a winning declaration has been determined via Layer Order, the Cascade won't even check Specificity or Order of Appearance for those declarations anymore. This is because Layers is a separate and higher ranked criterion of the Cascade.

Layers and the cascade. Layers and !important

With Cascade Layers at a lower level than origins & importance, layered declarations will continue to divide into the existing origin structure:

1. Transitions

2. User Agent !important
3. Author !important
4. Animations
5. Author
6. User
7. User Agent

So, taking our previous example:

```
@layer normalize { &hellip; }
@layer base { &hellip; }
@layer theme { &hellip; }
@layer content { &hellip; }
```

The precendce order becomes

1. Important Author Origin
    1. Important normalize layer
    2. Important base layer
    3. Important theme layer
    4. Important content layer
    5. Important unlayered styles
2. Animations
3. Normal Author Origin
    1. unlayered styles
    2. content layer
    3. theme layer
    4. base layer
    5. normalize layer

To use or not to use

When building a new project, this is the way to go, but the question I have is this: How do we make sure that content in layers works well with existing content that is not in layers?

If I understand it correctly content outside layers will be read last ina virtual unnamed layer, overriding content in all layers before it, so there may still be some significant work to add layers to existing content.

From my point of view, more research is necessary to see how difficult it will be to integrate layers with existing content and how easy they make to build new content.

More information

- CSS Cascade 5 specification
- Cascade layers explainer
- Cascade layers — MDN
- Cascade Layers — Una Kravets
- Hello, CSS Cascade Layers — Ahmad Shadeed