@layer

Baseline Widely available

The `@layer` CSS at-rule is used to declare a cascade layer and can also be used to define the order of precedence in case of multiple cascade layers.

Try it

CSS Demo: @layer

CSS HTML

```
@layer module, state;

@layer state {
  .alert {
    background-color: brown;
  }
  p {
    border: medium solid limegreen;
  }
}

@layer module {
```

OUTPUT

Beware of the zombies

Syntax

CSS

```
/* statement at-rules */
@layer layer-name;
@layer layer-name, layer-name, layer-name;

/* block at-rules */
@layer {rules}
@layer layer-name {rules}
```

where:

layer-name

Is the name of the cascade layer.

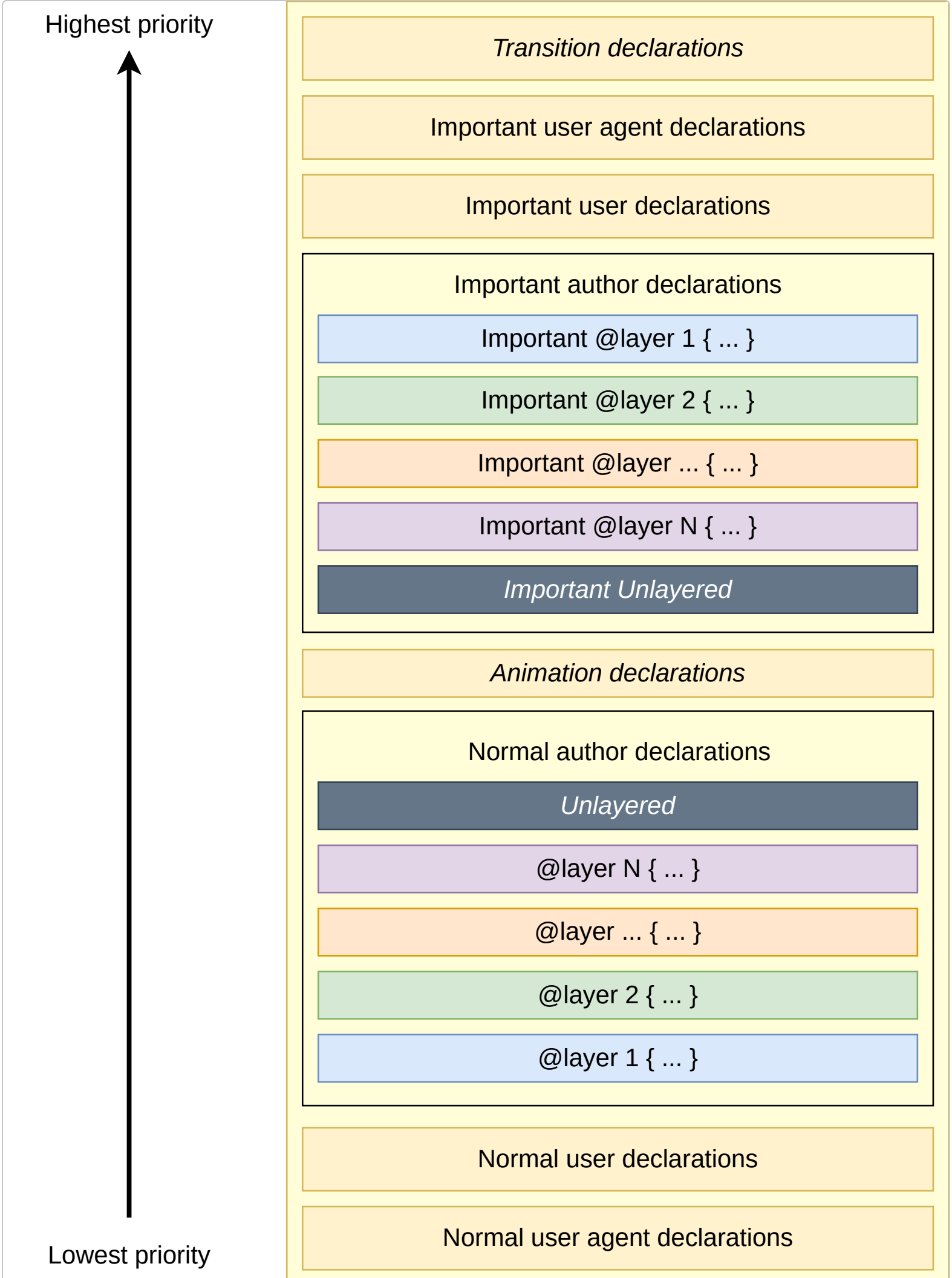
rules

Is the set of CSS rules in the cascade layer.

Description

Rules within a cascade layer cascade together, giving more control over the cascade to web developers. Styles that are not defined in a layer always override styles declared in named and anonymous layers.

The following diagram shows layer priorities where layers are declared in 1, 2, ..., N order.


Highest priority
Transition declarations
Important user agent declarations
Important user declarations
Important author declarations
Important @layer 1 { ... }
Important @layer 2 { ... }
Important @layer ... { ... }
Important @layer N { ... }
Important Unlayered
Animation declarations
Normal author declarations
Unlayered
@layer N { ... }
@layer ... { ... }
@layer 2 { ... }
@layer 1 { ... }
Normal user declarations
Normal user agent declarations
Lowest priority

As noted in the above diagram, *important declarations*, declarations with the `!important` flag, have priority over *normal declarations*, or regular declarations without the `!important` flag. The order of precedence among important rules is the inverse of normal rules. Transitions have the greatest precedence. Next in order of highest to lowest priority are the important user agent declarations, important user declarations, and important author declarations; in that order. Users can specify styles using browser preferences, operating system preferences, or browser extensions. Their important declarations take precedence over *author*, or *web developer* written, important declarations.

Within author styles, all important declarations within CSS layers take precedence over any important declarations declared outside of a layer, while all normal declarations within CSS layers have a lower priority than declarations declared outside of a layer. The declaration order matters. The first declared layer gets the lowest priority and the last declared layer gets the highest priority. However, the priority is reversed when the `!important` flag is used.

The `@layer` at-rule is used to create a cascade layer in one of three ways.

The first way is to use a `@layer` block at-rule to create a named cascade layer with the CSS rules for that layer inside, like so:

CSS

```css
@layer utilities {
  .padding-sm {
    padding: 0.5rem;
  }

  .padding-lg {
    padding: 0.8rem;
  }
}
```

The second way is to use a `@layer` statement at-rule to create one or more comma-separated named cascade layers without assigning any styles. This can be a single layer, as shown below:

CSS

```css
@layer utilities;
```

Multiple layers can be defined at once, as shown below:

CSS

```css
@layer theme, layout, utilities;
```

This is useful because the initial order in which layers are declared indicates which layer has precedence. As with declarations, the last layer to be listed will win if declarations are found in multiple layers. Therefore, with the preceding example, if a competing rule was found in `theme` and `utilities`, the one in `utilities` would win and be applied.

A rule in `utilities` would be applied *even if it has lower specificity* than the rule in `theme`. This is because once the layer order has been established, specificity and order of appearance are ignored. This enables using simpler CSS selectors because you do not have to ensure that a selector will have high enough specificity to override competing rules; all you need to ensure is that it appears in a later layer.

> **Note:** Having declared your layer names, thus setting their order, you can add CSS rules to the layer by re-declaring the name. The styles are then appended to the layer and the layer order will not be changed.

The third way is to create an unnamed layer using a `@layer` block at-rule without including a layer name. For example:

CSS

```
@layer {
  p {
    margin-block: 1rem;
  }
}
```

This creates an *anonymous cascade layer*. This layer functions in the same way as named layers; however, rules cannot be assigned to it later. The order of precedence for anonymous layers is the order in which layers are declared, named or not, and lower than the styles declared outside of a layer.

Another way to create a cascade layer is by using `@import`. In this case, the rules would be in the imported stylesheet. Remember that the `@import` at-rule must precede all other types of rules, except `@charset` and `@layer` rules.

CSS

```
@import "theme.css" layer(utilities);
```

Nesting layers

Layers may be nested. For example:

CSS

```
@layer framework {
  @layer layout {
  }
}
```

To append rules to the `layout` layer inside `framework`, join the two names with a `.`.

CSS

```
@layer framework.layout {
  p {
    margin-block: 1rem;
  }
}
```

Formal syntax

```
@layer =
  @layer <layer-name>? { <rule-list> }  |
  @layer <layer-name># ;

<layer-name> =
  <ident> [ '.' <ident> ]*
```

This syntax reflects the latest standard as per CSS Cascading and Inheritance Level 5. Not all browsers may have implemented every part. See Browser compatibility for support information.

Examples

Basic example

In the following example, two CSS rules are created. One for the `<p>` element outside of any layer and one inside a layer named `type` for `.box p`.

Without layers, the selector `.box p` would have the highest specificity, and therefore, the text `Hello, world!` will display in green. As the `type` layer comes before the anonymous layer created to hold non-layer content, the text will be purple.

Also notice the order. Even though we declare the non-layered style first, it's still applied *after* the layer styles.

HTML

HTML

```
<div class="box">
  <p>Hello, world!</p>
</div>
```

CSS

CSS

```css
p {
  color: rebeccapurple;
}

@layer type {
  .box p {
    font-weight: bold;
    font-size: 1.3em;
    color: green;
  }
}
```

Result

Hello, world!

Assigning rules to existing layers

In the following example, two layers are created with no rules applied, then CSS rules are applied to the two layers. The `base` layer defines a `color`, `border`, `font-size`, and `padding`. The `special` layer defines a different color. As `special` comes last when the layers were defined, the color it provides is used and the text is displayed using `rebeccapurple`. All of the other rules from `base` still apply.

HTML

HTML

```html
<div class="item">
  I am displayed in <code>color: rebeccapurple</code> because the
  <code>special</code> layer comes after the <code>base</code> layer. My green
  border, font-size, and padding come from the <code>base</code> layer.
</div>
```

CSS

CSS

```css
@layer base, special;

@layer special {
  .item {
    color: rebeccapurple;
  }
}

@layer base {
  .item {
    color: green;
    border: 5px solid green;
    font-size: 1.3em;
    padding: 0.5em;
  }
}
```

Result

I am displayed in `color: rebeccapurple` because the `special` layer comes after the `base` layer. My green border, font-size, and padding come from the `base` layer.

Specifications

Specification
CSS Cascading and Inheritance Level 5 # layering

Browser compatibility

Report problems with this compatibility data • View data on GitHub

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android	WebView on iOS
`@layer`	✓ 99	✓ 99	✓ 97	✓ 85	✓ 15.4	✓ 99	✓ 97	✓ 68	✓ 15.4	✓ 18	✓ 99	✓ 15.4

✓ Full support

See also

- `@import`
- `CSSLayerBlockRule`
- `CSSLayerStatementRule`
- `!important`
- `revert-layer`
- Introducing the CSS cascade
- Learn: Handling conflicts
- Learn: Cascade layers
- The future of CSS: Cascade layers on bram.us (2021)