

<https://drafts.csswg.org/css-cascade-4/#intro>

CSS Cascading and Inheritance Level 4

Editor's Draft, 8 June 2021

Specification Metadata

Copyright © 2021 W3C® (MIT, ERCIM, Keio, Beihang). W3C [liability](#), [trademark](#) and [permissive document license](#) rules apply.

Abstract

This CSS module describes how to collate style rules and assign values to all properties on all elements. By way of cascading and inheritance, values are propagated for all properties on all elements.

CSS 캐스 케이딩 및 상속 레벨 4

편집자 초안, 2021년 6월 8일

요약

이 문서는 CSS가 가지고 있는 속성값들과 요소들에 대한 규칙을 설명합니다. 계단식 및 상속을 통해 모든 요소의 모든 속성들이 서로 영향을 주고 받습니다.

예를 들어 `html` 태그는 태그 아래에 태그가 있고 CSS는 HTML의 구조처럼 계단식으로 적용된다는 것이다.

One of the fundamental design principles of CSS is [cascading](#), which allows several style sheets to influence the presentation of a document. When different declarations try to set a value for the same element/property combination, the conflicts must somehow be resolved.

CSS의 기본적인 디자인 원칙들 중 하나인 cascading(캐스케이딩, 계단식 배열)은 여러 스타일 시트가 문서(DOM = web page = html + css)에 영향을 준다는 것입니다.

즉, 여러가지 CSS 파일들이 한 가지의 HTML 태그에 영향을 준다는 것이다. 예를 들어 h1 태그에 대한 CSS 속성값을 각기 다르게 선언한 다른 두 개의 파일을 같이 넣는다면 어떻게 될까?

실험 결과 마지막 CSS 파일에 선언된 값을 기준으로 적용이 된다. 많은 것들이 적용된 웹페이지라면 이것이 막 뒤죽박죽 섞여 있으면 무엇이 마지막으로 적용되었는지 찾기가 매우 어려워진다. (테스트-코드-1)

3. Shorthand Properties

Some properties are shorthand properties, meaning that they allow authors to specify the values of several properties with a single property. A [shorthand property](#) sets all of its longhand sub-properties, exactly as if expanded in place.

When values are omitted from a [shorthand](#) form, unless otherwise defined, each “missing” [sub-property](#) is assigned its [initial value](#).

This means that a [shorthand](#) property declaration always sets *all* of its [sub-properties](#), even those that are not explicitly set. Carelessly used, this might result in inadvertently resetting some sub-properties. Carefully used, a shorthand can guarantee a “blank slate” by resetting sub-properties inadvertently cascaded from other sources.

For example, writing [background: green](#) rather than [background-color: green](#) ensures that the background color overrides any earlier declarations that might have set the background to an image with [background-image](#).

For example, the CSS Level 1 [font](#) property is a [shorthand](#) property for setting [font-style](#), [font-variant](#), [font-weight](#), [font-size](#), [line-height](#), and [font-family](#) all at once. The multiple declarations of this example:

```
h1 {  
    font-weight: bold;  
    font-size: 12pt;  
    line-height: 14pt;  
    font-family: Helvetica;  
    font-variant: normal;  
    font-style: normal;  
}
```

can therefore be rewritten as

```
h1 { font: bold 12pt/14pt Helvetica }
```

As more [font sub-properties](#) are introduced into CSS, the shorthand declaration resets those to their initial values as well.

In some cases, a [shorthand](#) might have different syntax or special keywords that don't directly correspond to values of its [sub-properties](#) (In such cases, the shorthand will explicitly define the expansion of its values.)

In other cases, a property might be a reset-only sub-property of the shorthand: Like other [sub-properties](#), it is reset to its initial value by the shorthand when unspecified, but the shorthand might not include syntax to set the sub-property to any of its other values. For example, the [border](#) shorthand resets [border-image](#) to its initial value of none, but has no syntax to set it to anything else. [\[css-backgrounds-3\]](#)

If a [shorthand](#) is specified as one of the [CSS-wide keywords](#) [\[css-values-3\]](#), it sets all of its [sub-properties](#) to that keyword, including any that are [reset-only sub-properties](#). (Note that these keywords cannot be combined with other values in a single declaration, not even in a shorthand.)

Declaring a [shorthand](#) property to be !important is equivalent to declaring all of its [sub-properties](#) to be !important.

CSS 속성을 하나하나 쓸 수도 있지만 동일한 범주내의 속성의 경우 줄여서 쓸 수 있다.
(테스트-코드-2)

Properties sometimes change names after being supported for a while, such as vendor-prefixed properties being standardized. The original name still needs to be supported for compatibility reasons, but the new name is preferred. To accomplish this, CSS defines two different ways of “aliasing” old syntax to new syntax.

CSS 속성값들의 이름들은 vendor-prefixed properties가 표준화 될 때 종종 바뀝니다. 기존의 표준 속성값의 이름은 호환성을 이유로 필요하지만 새로운 이름이 선호되기도 합니다. 이러한 경우 CSS의 정의는 두 가지 방식으로 결정되는데 이를 앤리어싱이라고 합니다.

<https://www.thoughtco.com/css-vendor-prefixes-3466867> 참고

즉, CSS의 속성값의 이름은 시간이 지남에 따라 브라우저 제조사별로 특정한 속성값의 이름을 바꾸거나 새로운 속성값을 추가하는 경우가 있는데 이러한 경우에 기존의 이름과 새로운 이름을 유지하는 것이 바로 앤리어싱입니다.

호환을 위해 고려할 수 있지만 일반적으로 구식 브라우저에 맞춰져 있기 때문에 현대에는 사용할 일이 거의 없습니다.

`all:initial;`

`all:inherit;`

`all:unset;`

`all initial`은 기존의 CSS 값을 초기화시키는데 코드가 사라지는 것은 아니다. 각 요소마다 적용할 수 있는데 다 만들고 어디가 이상할 때 파트별로 볼 수 있으니 하나씩 지우면서 살펴볼 때 편할 것 같다.

`all:inherit;`은 기존의 CSS 값을 상속값으로 변경한다고 되어 있다. 이 말을 이해하기 위해서는 지정값과 상속값의 개념에 대해 알아야 한다.

지정값은 `px` 값을 생각하면 된다. 폰트의 크기를 정할 때 `px` 크기로 정하면 어느 모니터에서도 해당 `px` 크기를 유지한다. 다만 문제는 `px`로 고정값(지정값)을 만들어버리면 휴대폰에서는 깨지게 된다.

따라서 html의 `body`부분의 가로 세로 길이를 기기의 화면 크기에 따라 `px` 단위로 딱딱 잘라서 개발하려면 일반적인 포맷인 1980x1080부터 1366x768, 1440x900, 360x640 등등 휴대폰, 태블릿, pc 모니터에 따라 각기 다른 CSS 값을 지정해야 합니다.

그래서 현대에는 반응형 웹페이지 디자인을 사용합니다. 아래는 반응형 웹페이지에 대한 설명 링크

https://www.w3schools.com/html/html_responsive.asp

<https://www.browserstack.com/guide/how-to-create-responsive-website>