






CSS Cascade Layers


CSS Cascade Layers

@layer

 **Baseline 2022** NEWLY AVAILABLE

    ^

Since March 2022, this feature works across the latest devices and browser versions. This feature might not work in older devices or browsers.

[Learn more](#) [See full compatibility](#)  [Report feedback](#)

@layer - CSS: Cascading Styl x

Cascade Layers | CSS-Tricks x

AI sketch x

←

→

↺

⚠ 주의 요함

ec2-13-124-58-80.ap-northeast-2.compute.amazonaws.com

☆

□

전

⋮

img.chakra-image.css-1t80rzi

266 × 187

ACCESSIBILITY


Name

Role

Keyboard-focusable

image

ⓧ



텍스트 스케치

ⓘ DevTools is now available in Korean!

Always match Chrome's language

Switch DevTools to Korean

Don't show again

🔍

📄

Elements

Console

Sources

Network

Performance

Memory

>>

⚙️

⋮

✕

html

body.chakra-ui-light

div#root

span#__chakra_env

Styles

Computed

Layout

Event Listeners

DOM Breakpoints

Properties

Accessibility

Filter

:hov

.cls

+

🔍

📄

element.style {

}

[hidden] {

display: none !important;

}

* {

font-family: "Pretendard Variable", Pretendard, sans-serif;

}

*, ::before, ::after {

border-color: ▾ ■ var(--chakra-colors-chakra-border-color);

}

:where(*) {

border-width: ▸ 0px;

border-style: ▸ solid;

box-sizing: border-box;

overflow-wrap: break-word;

}

span[Attributes Style] {

display: none;

}

Inherited from body.chakra-ui-light

body {

font-family: var(--chakra-fonts-body);

color: ▢ var(--chakra-colors-chakra-body-text);

background: ▸ ■ var(--chakra-colors-chakra-body-bg);

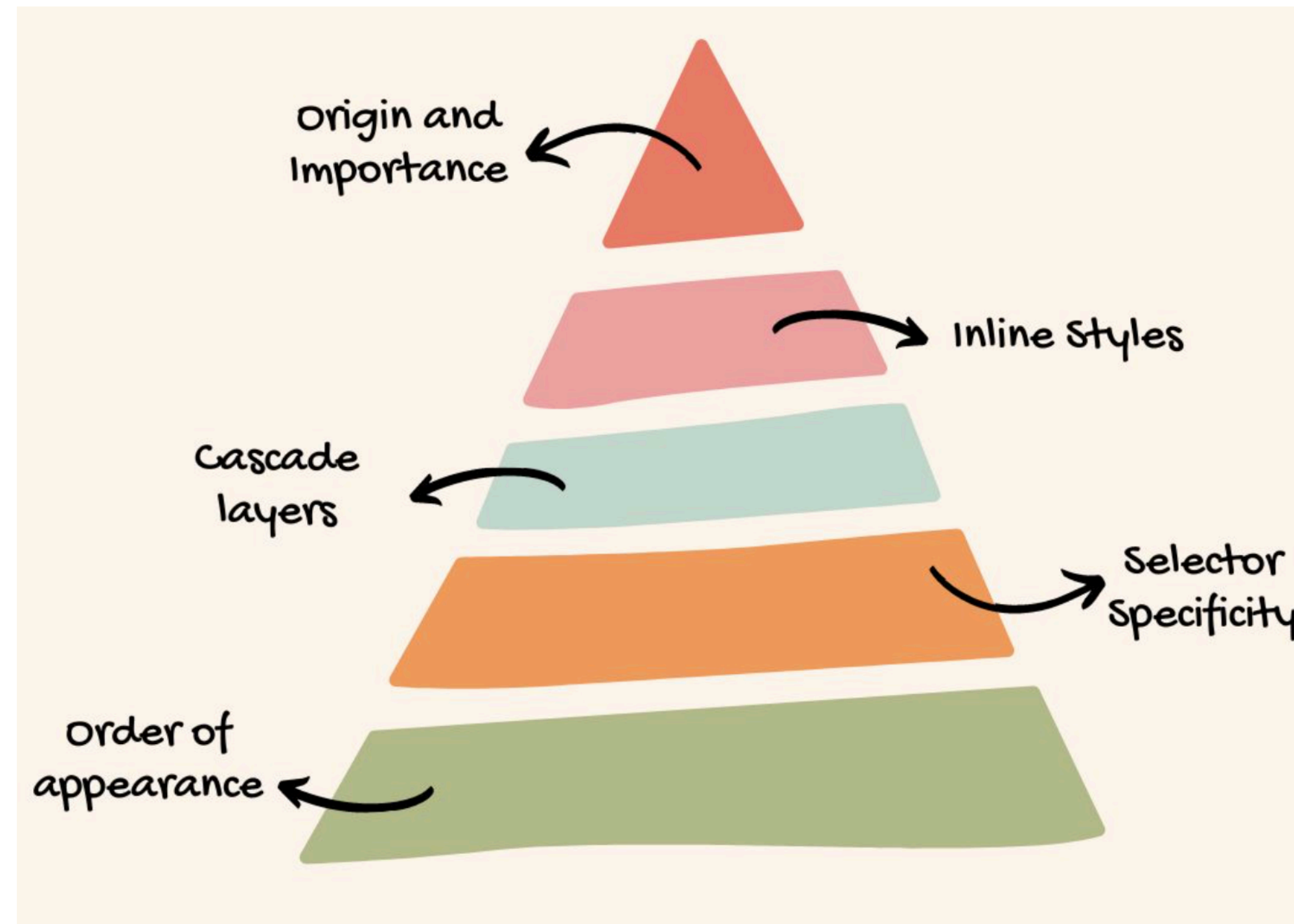
transition-property: background-color;

transition-duration: var(--chakra-transition-duration-normal);

line-height: var(--chakra-lineHeights-base);

}

CSS Cascade



여러 source로부터의 css에서 정의된 여러 스타일 값들 중에서 요소에 적용할 스타일을 결정하는 방법

CSS Cascade

1. **Relevance:** Find all the declaration blocks with a selector match for each element.
2. **Importance:** Sort rules based on whether they are normal or important. Important styles are those that have the `!important` flag set.
3. **Origin:** Within each of the two importance buckets, sort rules by author, user, or user-agent origin.
4. **Cascade layers:** Within each of the six origin importance buckets, sort by cascade layer. The layer order for normal declarations is from the first layer created to the last, followed by unlayered normal styles. This order is inverted for important styles, with unlayered important styles having the lowest precedence.

CSS Cascade

5. **Specificity:** For competing styles in the origin layer with precedence, sort declarations by [specificity](#).
6. **Scoping proximity:** When two selectors in the origin layer with precedence have the same specificity, the property value within scoped rules with the smallest number of hops up the DOM hierarchy to the scope root wins. See [How `@scope` conflicts are resolved](#) for more details and an example.
7. **Order of appearance:** When two selectors in the origin layer with precedence have the same specificity and scope proximity, the property value from the last declared selector with the highest specificity wins.

CSS Cascade

Specificity

- 한 요소와 가장 관련성이 높은 CSS 선언을 결정하기 위해 사용하는 알고리즘
- selector의 “가중치(weight)”를 계산
- <id column> - <class column> - <type column> 형식

CSS Cascade

Specificity

1. id column

(예: #example), 1-0-0

2. class column

(예: .myClass, [type="radio"], :hover), 0-1-0

3. type column

(예: p, h1, [type="radio"], ::before, ::placeholder), 0-0-1

4. no value

(예: *, :where(), ::before, ::placeholder), 0-0-0

CSS Cascade

Specificity

CSS

```
#myElement {  
  color: green; /* 1-0-0 - WINS!! */  
}  
.bodyClass .sectionClass .parentClass [id="myElement"] {  
  color: yellow; /* 0-4-0 */  
}
```

Example 1

CSS

```
#myElement {  
  color: yellow; /* 1-0-0 */  
}  
#myApp [id="myElement"] {  
  color: green; /* 1-1-0 - WINS!! */  
}
```

Example 2

CSS Cascade

Order (low to high)	Origin	Importance
1	user-agent (browser)	normal
2	user	normal
3	author (developer)	normal
4	CSS @keyframe animations	
5	author (developer)	!important
6	user	!important
7	user-agent (browser)	!important
8	CSS transitions	

CSS Cascade

User agent styles

Author styles

Interactive editor

```
:where(a.author) {  
  text-decoration: overline;  
  color: red;  
}
```

```
<p><a href="https://example.org">User agent styles</a></p>  
<p><a href="https://example.org" class="author">Author styles</a></p>
```

User-agent → a:any-link 0-1-1
Author → :where(a.author) 0-0-0

CSS Cascade Layers

배경

- 여러 source의 style sheets는 단일 출처로 cascading
- 대규모 코드 베이스 작업 시 specificity 충돌 발생
- 팀마다 다른 해결 방법론
- !important로는 근본적인 해결 불가능

CSS Cascade Layers

- cascade layer는 3개 origin 각각의 'sub-origin'을 만드는 것
- Layer 우선순위는 '레이어 생성 시점' 기준
 - Normal origin의 경우,
가장 먼저 생성 < 두번째 생성 < ... < 마지막 생성 < unlayered styles(가장 높음)
 - Important origin의 경우,
unlayered styles < 마지막 생성 < ... < 가장 먼저 생성

Cascade Layer 생성

@layer at-rule statement

CSS



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


Cascade Layer 생성

@layer at-rule block

CSS

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

```
/* unlayered styles */
body {
  color: #333;
}

/* creates the first layer: `layout` */
@layer layout {
  main {
    display: grid;
  }
}

/* creates the second layer: an unnamed, anonymous layer */
@layer {
  body {
    margin: 0;
  }
}

/* creates the third and fourth layers: `theme` and `utilities` */
@layer theme, layout, utilities;

/* adds styles to the already existing `layout` layer */
@layer layout {
  main {
    color: #000;
  }
}

/* creates the fifth layer: an unnamed, anonymous layer */
@layer {
  body {
    margin: 1vw;
  }
}
```

```
@media (min-width: 50em) {  
  @layer site;  
}  
  
@layer page {  
  h1 {  
    text-decoration: overline;  
    color: red;  
  }  
}  
  
@layer site {  
  h1 {  
    text-decoration: underline;  
    color: green;  
  }  
}
```

```
<h1>Is this heading underlined?</h1>
```

Is this heading underlined?

넓은 화면

Is this heading underlined?

좁은 화면

Cascade Layer 생성

@import at-rule

CSS



```
@import url("components-lib.css") layer(components);  
@import url("dialog.css") layer(components.dialog);  
@import url("marketing.css") layer();
```

Cascade Layer pitfalls

레이어 계층 구조의 복잡한 구성, 레이어의 잘못된 관리

⇒ 개발자들 사이에 혼란

⇒ 스타일이 의도하지 않은 영역으로 번지거나 의도한 대로 적용x

Cascade Layer pitfalls 벗어나기

- 계층 구조를 단순화하여 명확하고 이해하기 쉬운 구조를 유지
- 논리적인 레이어 정의
- 구체성과 유연성 사이의 균형을 찾기
- 레이어 계층 구조와 그 근거를 적절히 문서화
- 브라우저 호환성 테스트

END