

# CSS 기본

CSS Basic

CSS란 무엇이며, 어떻게 사용하는 것인지?

# CSS란?

Cascading Style Sheet

마크업 언어(HTML)가 실제 표시되는 방법을 기술하는 언어  
레이아웃과 스타일을 정의할 때 사용

HTML과 CSS를 분리해서 사용해야 하는 이유는?

# 과거의 HTML

혼돈! 파괴! 망각각!

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  </head>
  <body bgcolor="Azure">
    <font size="4" face="arial" color="DarkBlue">Chaos! Destruction! Oblivion!</font>
    <hr size="1" width="100%" color="DarkCyan">
    <p font-size="20px">CSS가 없는 혼돈의 세계입니다</p>
    <br>
    <hr align="right" width="40%" color="DarkSlateGray">
    <p align="right">여러분은 이렇게 사용하시면 안됩니다</p>
  </body>
</html>
```

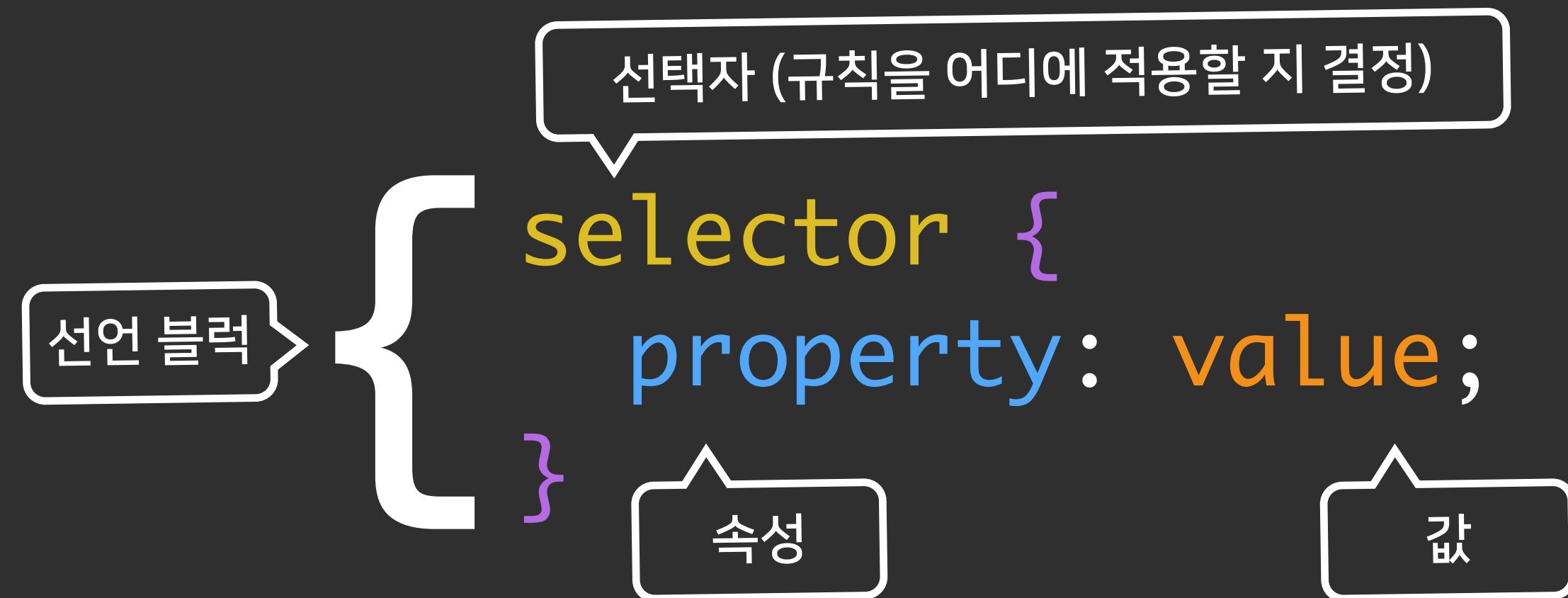
HTML에는 스타일을 제외한, 문서의 구조만이 명확히 나타나야 함

과거 브라우저에서 사용하던 font, center 등의 태그는 HTML5에서는 더 이상 사용하지 않습니다

[http://www.w3schools.com/html/html5\\_intro.asp](http://www.w3schools.com/html/html5_intro.asp)

# CSS문법

## CSS Syntax Basic



# CSS문법

## CSS Syntax Basic - Example

```
#body-title {  
    font-size: 14px;  
    font-weight: bold;  
    color: DarkSlateGrey;  
}
```

HTML Color Names

[http://www.w3schools.com/colors/colors\\_names.asp](http://www.w3schools.com/colors/colors_names.asp)

# CSS사용법

내부 스타일 시트 (Internal Style Sheet)

```
<html lang="en">  
<head>  
  <style type="text/css">  
    #body-title {  
      font-size: 14px;  
      font-weight: bold;  
      color: DarkSlateGrey;  
    }  
  </style>  
</head>  
<body>  
  <p id="body-title">Internal Style Sheet</p>  
</body>  
</html>
```

head 안쪽, style태그 내부에 작성

# CSS사용법

## 인라인 스타일 시트 (Inline Style Sheet)

```
<html lang="en">
<head>
</head>
<body>
  <p id="body-title" style="font-size:14px; font-weight: bold; color: DarkSlateGrey">Inline Style Sheet</p>
</body>
</html>
```

사용할 요소의 style속성에 정의

인라인 스타일은 내용과 스타일이 분리되지 않으므로 권장되지 않습니다

# CSS사용법

외부 스타일 시트 (External Style Sheet)

```
<html lang="en">
<head>
  <link rel="stylesheet" href="style.css">
</head>
<body>
  <p id="body-title">External Style Sheet</p>
</body>
</html>
```

link태그를 사용, href속성에 경로를 입력

link태그를 사용하여 외부 CSS파일을 HTML문서에 연결합니다



# CSS 선택자

CSS Selector

CSS를 적용할 요소

# CSS선택자

Universal Selector (전체 선택자)

별표 기호 (Asterisk)

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

HTML페이지 내부의 모든 요소에 같은 CSS속성을 적용합니다.

따라서 margin이나 padding값을 초기화하는 등, 기본값을 정할 때 주로 사용합니다.

다만 문서의 모든 요소를 읽기 때문에 페이지 로딩시간이 길어질 수 있으니 자주 사용하는 것은 좋지 않습니다.

# CSS선택자

Tag Selector (태그 선택자)

HTML 태그명

```
h1 {  
  color: red;  
}
```

HTML 태그명

```
p {  
  color: gray;  
  margin-bottom: 10px;  
}
```

해당하는 모든 HTML태그 요소를 지정합니다.

# CSS선택자

Class Selector (클래스 선택자)

마침표 기호

CSS

```
.section {  
  color: #333;  
  margin-bottom: 40px;  
}  
p.section-title {  
  font-size: 18px;  
}  
p.section-content {  
  font-size: 13px;  
  line-height: 13px;  
  color: #999;  
}
```

앞에 TAG명 입력 가능

HTML

```
<body>  
  <div class="section">  
    <p class="section-title">Lorem ipsum dolor sit amet.</p>  
    <p class="section-content">Lorem ipsum dolor sit amet</p>  
  </div>  
  
  <div class="section">  
    <p class="section-title">Lorem ipsum dolor sit amet.</p>  
    <p class="section-content">Lorem ipsum dolor sit amet</p>  
  </div>  
</body>
```

CSS에서는 마침표 기호로 나타내며, HTML에서는 주어진 값을 class속성값으로 가진 요소를 선택합니다.

마침표 앞에 태그를 붙여주면 범위는 지정한 태그에 한합니다.

클래스는 쉼표(,)로 구분하여 동시에 여러개의 값을 지정할 수 있습니다.

# CSS선택자

## ID Selector (ID 선택자)

# 기호

CSS

```
#index-title {  
  font-size: 18px;  
}  
p#index-description {  
  font-size: 12px;  
  color: #999;  
}
```

앞에 TAG명 입력 가능

HTML

```
<body>  
  <h3 id="index-title">Lorem ipsum dolor sit.</h3>  
  <p id="index-description">Lorem ipsum doloro?</p>  
</body>
```

CSS에서는 #기호로 나타내며, HTML에서는 주어진 값을 id속성값으로 가진 요소를 선택합니다.

HTML에서 id값은 오직 하나만 존재해야 합니다.

클래스 선택자와 같이 앞에 TAG명을 입력할 수 있습니다.

ID선택자의 우선순위가 Class선택자의 우선순위보다 높으므로, 같은 속성에 서로 다른 값을 지정할 경우 ID선택자의 값이 적용됩니다.

# CSS선택자

Chain Selector (체인 선택자)

CSS

```
#index-title {  
  font-size: 18px;  
}  
#index-title.header-title {  
  font-weight: bold;  
}  
.body-text {  
  font-size: 12px;  
}  
.body-text.description {  
  color: #999;  
}
```

둘 이상 요소에 같은 스타일 적용

HTML

```
<body>  
  <h3 id="index-title" class="header-title">  
    Lorem ipsum dolor sit.</h3>  
  <p class="body-text description">  
    Lorem ipsum doloro?</p>  
</body>
```

한 요소에 아이디와 클래스들, 또는 여러 클래스가 적용되어 있을 경우에 사용합니다.

# CSS선택자

Group Selector (그룹 선택자)

CSS

```
#index-title {  
  font-size: 18px;  
}  
p#index-description {  
  font-size: 12px;  
  color: #999;  
}  
#index-title, #index-description {  
  text-align: center;  
}
```

둘 이상 요소에 같은 스타일 적용

HTML

```
<body>  
  <h3 id="index-title">Lorem ipsum dolor sit.</h3>  
  <p id="index-description">Lorem ipsum doloro?</p>  
</body>
```

여러 선택자에 같은 스타일을 적용하는 경우, 쉼표로 구분해 선택자를 나열해 사용합니다

# CSS선택자

## Combinator Selector (복합 선택자)

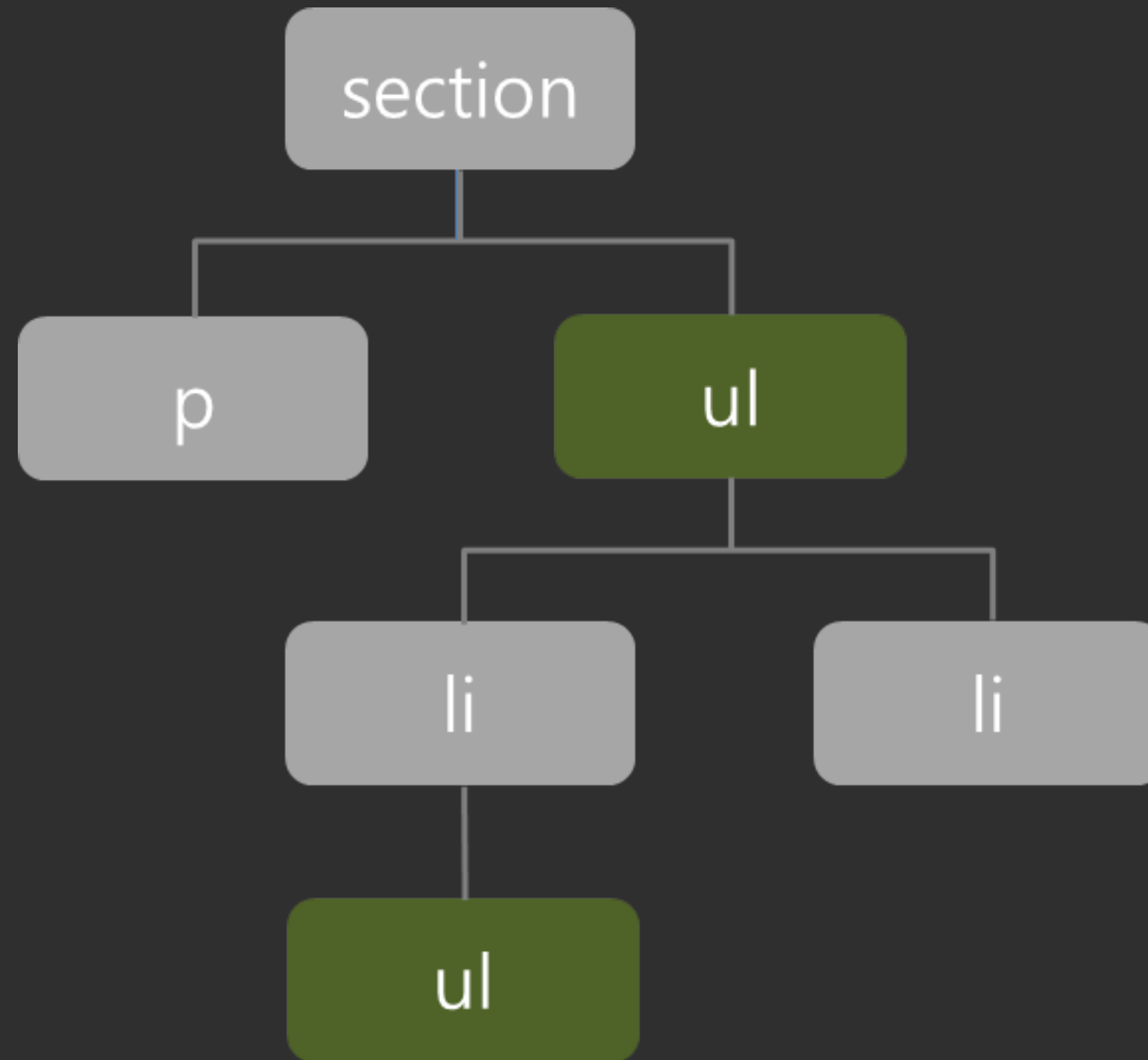
하위선택자와 자식선택자 (Descendant and Child)

하위 선택자 (descendant selector)

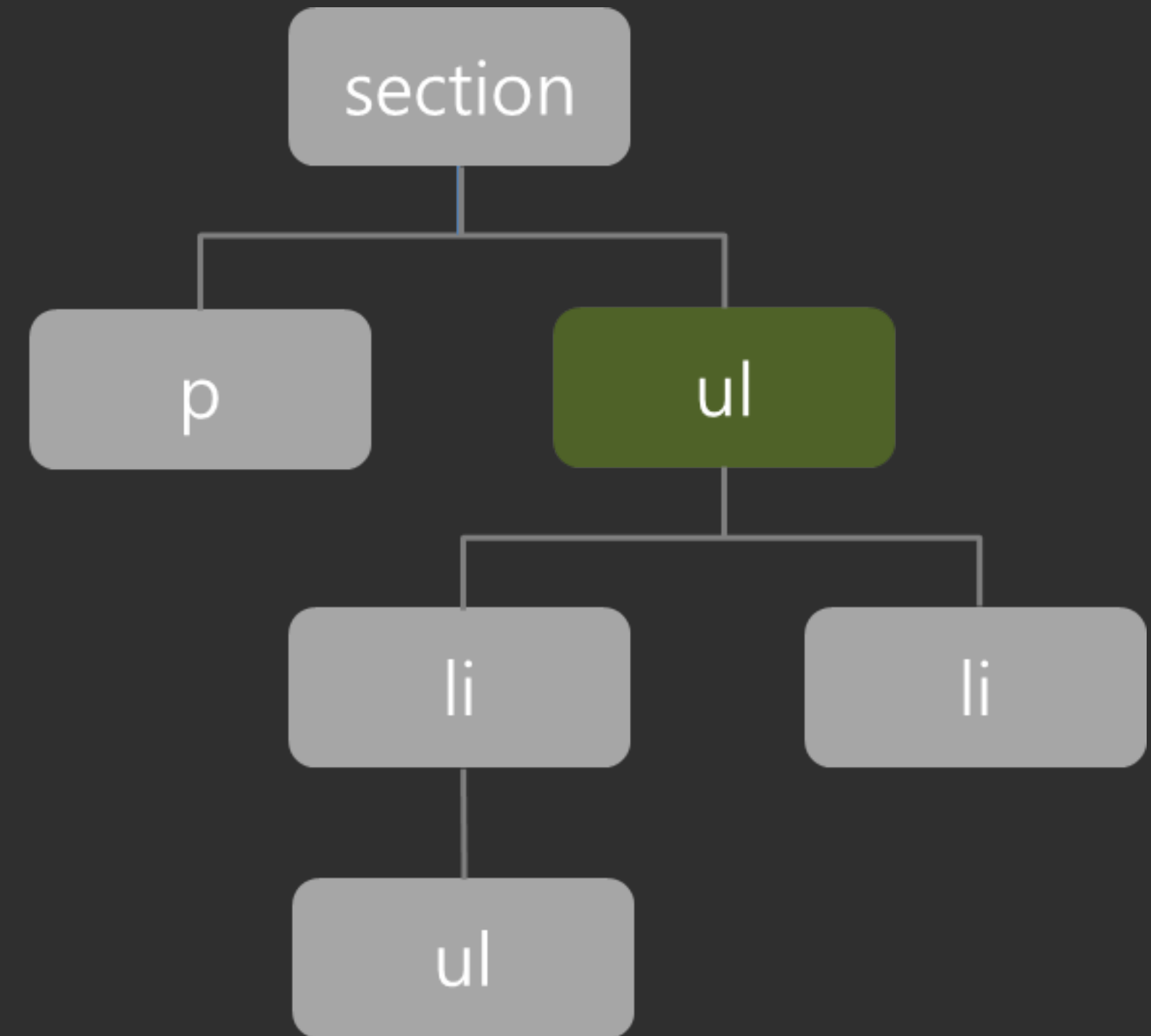
```
section ul {  
  border: 1px solid black;  
}  
section > ul {  
  border: 1px solid black;  
}
```

자식 선택자 (Child selector)

### 하위 선택자



### 자식 선택자



포함 관계를 가지는 태그들 사이에서, 포함하는 요소는 '부모 요소', 포함되는 요소는 '자식 요소'라고 합니다.

하위 선택자는 부모요소에 포함된 '모든' 하위 요소를 지정하며,

자식 선택자는 부모요소의 '바로 아래' 자식 요소만을 지정합니다.



# CSS선택자

## Combinator Selector (복합 선택자)

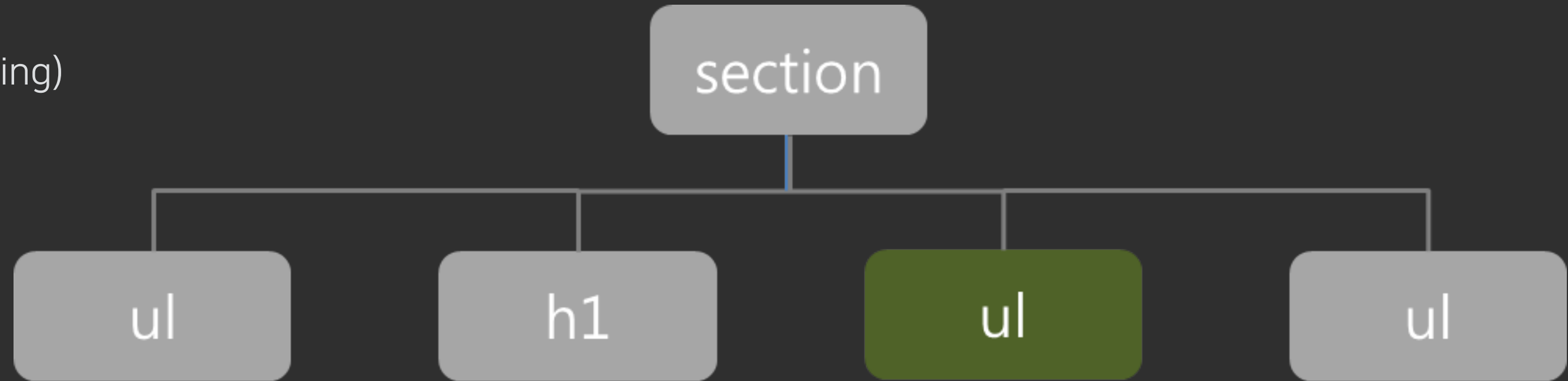
인접 형제 선택자와 일반 형제 선택자 (Adjacent Sibling & General Sibling)

### 인접 형제 선택자

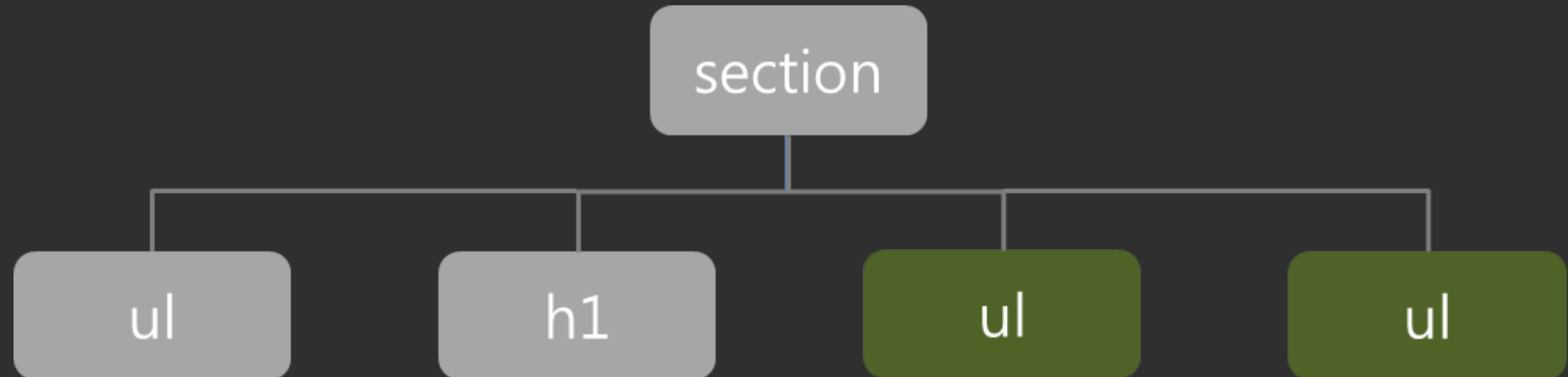
```
h1 + ul {  
  background: LightBlue;  
  color: DarkBlue;  
}  
h1 ~ ul {  
  background: LightBlue;  
  color: DarkBlue;  
}
```

### 일반 형제 선택자

### 인접 형제 선택자



### 일반 형제 선택자



같은 부모 요소를 가지는 요소들은 '형제 관계' 라고 부릅니다.

이 때, 먼저 나오는 요소를 '형 요소', 나중에 나오는 요소는 '동생 요소'라 합니다. (부모 요소 내부에서 보다 위 줄에 쓰여진 것을 의미합니다)

두 선택자 모두 형 요소에는 적용되지 않으며, 인접 형제 선택자는 조건을 충족하는 '첫 번째' 동생 요소만을 지정하며, 일반 형제 선택자는 조건을 충족하는 '모든' 동생 요소를 지정합니다.

# CSS선택자

## 속성 선택자 (Attribute Selector)

태그 내의 속성에 따름

패턴	의미	예제
E[attr]	'attr'속성이 포함된 요소 E	<E attr>Lorem</E>
E[attr="val"]	'attr'속성의 값이 'val'인 요소 E	<E attr="val">Lorem</E>
E[attr~="val"]	'attr'속성의 값에 'val'이 포함되는 요소 E (공백으로 분리된 값이 일치해야 함)	<E attr="val">Lorem</E> <E attr="item val">Lorem</E> <E attr="item-val">Lorem</E>
E[attr = "val"]	'attr'속성의 값에 'val'이 포함되거나 (공백분리), 'val-'로 시작하는 요소 E	<E attr="val">Lorem</E> <E attr="val-num3">Lorem</E> <E attr="value">Lorem</E>
E[attr^="val"]	'attr'속성의 값이 'val'로 시작하는 요소 E	<E attr="val">Lorem</E> <E attr="value">Lorem</E>
E[attr\$="val"]	'attr'속성의 값이 'val'로 끝나는 요소 E	<E attr="val">Lorem</E> <E attr="item-val">Lorem</E>
E[attr*="val"]	'attr'속성의 값에 'val'이 포함되는 요소 E (공백이나 Dash(-)에 영향받지 않음)	<E attr="val">Lorem</E> <E attr="many itemval">Lorem</E> <E attr="item-val">Lorem</E>

# CSS선택자

가상 클래스 선택자 (Pseudo-Classes Selector), 가상 엘리먼트 선택자 (Pseudo-Elements Selector)

HTML소스에는 존재하지 않지만 필요에 의해 가상의 선택자를 지정

패턴	의미
E:link	방문하지 않은 링크 E
E:visited	방문한 링크 E
E:active	E 요소에 마우스 클릭 또는 키보드 엔터가 눌린 동안
E:hover	E 요소에 마우스가 올라가 있는 동안
E:focus	E 요소에 포커스가 머물러 있는 동안
E::first-line	E 요소의 첫 번째 라인
E::first-letter	E 요소의 첫 번째 문자
E::before	E 요소의 시작 지점에 생성된 요소
E::after	E 요소의 끝 지점에 생성된 요소

# CSS 우선순위

CSS Cascading

스타일 적용의 우선순위를 알아봅니다

# CSS스타일 적용 우선순위

특정도(specify)값이 높은 순서대로 적용됩니다  
특정도는 가장 구체적인 값을 의미합니다

## 특정도 계산식

스타일	특정도
Inline (인라인 스타일)	A
ID Selector (ID 선택자)	B
Class Selector (클래스 선택자)	C
TAG Selector (태그 선택자)	D

특정도는 CSS구문에서  
해당하는 스타일의 수 \* 특정도 값을 모두 더한 값입니다.

클래스 선택자가 3개 존재하며, 태그선택자가 있는 CSS구문이라면

ex) p.wrap.item>.active

클래스선택자수(3) \* 클래스선택자의 특정도(C) + 태그선택자수(1) \* 태그 선택자의 특정도(D) = 3C,1D의 특정도를 가지게 됩니다.

## 예제)

CSS 구문	특정도
p { color: gray; }	1D
p:first-line { color: black; }	2D
.wrap { color: black; }	1C
p.wrap { color: black; }	1C,1D
p.wrap>.item { color: black; }	2C,1D
#wrap { color: black; }	1B
p#wrap { color: black; }	1B,1D
<!-- HTML --> <p style="color: black;">Example</p>	1A

# 우선순위 강제적용

## 인라인 스타일과 important

!important는 절대적인 우선순위를 가집니다

### 특정도 계산식

스타일	특정도
important	Absolute
Inline (인라인 스타일)	A
ID Selector (ID 선택자)	B
Class Selector (클래스 선택자)	C
TAG Selector (태그 선택자)	D

```
p {  
  font-size: 30px !important;  
  color: green !important;  
}
```

CSS

```
<p style="font-size: 10px; color: green;">important는 모든걸 무시하지</p>
```

HTML

important 쓴 것을 잊어버리면 나중에 왜 안되는지 헤매게 됨

!important값은 해당하는 요소에 지정된 어떤 특정도 무시하고 가장 우선순위로 적용됩니다.  
이는 테스트시에는 유용할 수 있지만, 추후 유지보수나 전체적 스타일 흐름을 방해하기 때문에 이용하지 않는 것이 좋습니다.

# CSS 서체

CSS Typography

CSS로 서체에 스타일을 지정합니다

# 서체 색상

Color

```
h1 {  
  color: #ff0000;  
}
```

color속성은 글자 색을 지정합니다.

HEX코드, rgb(), rgba(), 색상명이 올 수 있습니다.



# 서체 지정

Font family

```
body {  
    font-family: “돋움”, dotum, “굴림”, gulim, arial, helvetica, sans-serif;  
}
```

웹 페이지를 방문한 사용자가 없을 경우를 대비해, 다양한 서체들을 선언해놓습니다.

순서대로 해당 서체가 없을 경우 다음 서체가 사용자의 컴퓨터에 설치되어 있는지 확인 후, 순서 중 가장 먼저 찾은 폰트를 사용하게 됩니다.

# 서체의 종류

Font types

## Serif

글자에 꺾쇠가 붙어있는 서체, 인쇄물에 많이 사용  
영어 서체에는 Georgia, Times New Roman  
한글 서체에는 바탕체, 궁서체, 명조체가 있다

Python, 파이썬

## Sans-Serif

Sans는 “없음”을 뜻하는 프랑스어.  
Serif가 없는 글시체를 말함  
영어 서체에는 Arial, Helvetica  
한글 서체에는 돋움, 굴림, 나눔 고딕등이 있다.

Python, 파이썬

## Monospace

고정 폭 서체  
글자들 간 구분이 쉬워야 하는  
프로그래밍 코드를 나타낼 때 주로 사용

Python, 파이썬

## Cursive

커브가 많이 들어간 서체  
필기체를 말함  
가독성이 떨어지니 일부분에만 사용하는 것이 좋다

Python, 파이썬

# 글자 크기

Font size

```
body {  
    font-size: 14px;  
}  
h1 {  
    font-size: 28px(2em);  
}
```

\*\* em은 부모 요소로부터의 비율입니다.

부모(body)의 font-size가 14px이며, h1의 font-size가 2em이면  $14\text{px} \times 2.0 = 28\text{px}$ 이 적용됩니다.

# 글자 스타일

Font style

```
p {  
    font-style: italic;  
}
```

italic과 oblique는 둘 다 기울임꼴을 나타내지만, italic은 별도의 기울어진 폰트가 있을 경우 해당 폰트를 사용합니다.  
inherit은 상위 요소의 font-style을 물려받아 나타냅니다.

# 글자 굵기

Font weight

```
p {  
  font-weight: bold;  
}  
p {  
  font-weight: 700;  
}
```

lighter  
normal  
bold  
bolder  
inherit

<400  
400  
700  
>700

lighter와 bolder는, 해당 서체의 Lighter또는 Bolder서체가 있어야 표현되며,  
해당하는 서체가 없을 경우 normal, bold와 동일 굵기로 나타납니다.

# 줄 간격

Line height

```
p {  
  line-height: 1.5;  
}
```

숫자만 입력할 경우 해당 font-size  
에 곱한 값이 줄 간격이 됩니다

\*\* font-size가 px로 고정되어 있다면, line-height에도 고정된 px을 사용할 수 있습니다.

# 문자 꾸미기

text-decoration

```
p.item1 {  
  text-decoration: none;  
}  
p.item2 {  
  text-decoration: underline;  
}  
p.item3 {  
  text-decoration: overline;  
}  
p.item4 {  
  text-decoration: line-through;  
}
```

밑줄

윗줄

취소선

# 문자 정렬

text-align

```
p.item1 {  
    text-align: left;  
}  
p.item2 {  
    text-align: center;  
}  
p.item3 {  
    text-align: right;  
}  
p.item4 {  
    text-align: justify;  
}
```

양쪽정렬

\*\* justify는 우측 끝 부분을 깔끔하게 양쪽 정렬해주지만, 줄의 내부 간격이 뒤틀리므로 잘 사용하지 않습니다.



# 문자 들여쓰기

text-indent

```
p {  
  text-indent: 1em;  
}
```

-값일 경우, 내어쓰기

# 대소문자 변환

text-transform

```
p.item1 {  
  text-transform: capitalize;  
}  
p.item2 {  
  text-transform: uppercase;  
}  
p.item3 {  
  text-transform: lowercase;  
}
```

첫 글자만 대문자로

모두 대문자로

모두 소문자로

당연히 영문에만 적용됩니다.

# 자간

letter-spacing

```
p {  
  letter-spacing: 5px;  
}
```

각 글자간의 간격

# 단어 간격

word-spacing

글자가 아닌, 단어간의 간격을 조절합니다

```
p {  
  word-spacing: 5px;  
}
```

각 단어간의 간격

# 요소간 수직 정렬

## vertical-align

박스 내에서의 수직 정렬이 아닌, 나란히 오는 인라인 요소간의 정렬

```
<div>
  <strong>baseline : </strong>
  <span class="ori">align</span>
  <span class="vitem item1">Text</span>
</div>

<div>
  <strong>sub : </strong>
  <span class="ori">align</span>
  <span class="vitem item2">Text</span>
</div>

<div>
  <strong>super : </strong>
  <span class="ori">align</span>
  <span class="vitem item3">Text</span>
</div>

<div>
  <strong>top : </strong>
  <span class="ori">align</span>
  <span class="vitem item4">Text</span>
</div>

<div>
  <strong>text-top : </strong>
  <span class="ori">align</span>
  <span class="vitem item5">Text</span>
</div>

<div>
  <strong>bottom : </strong>
  <span class="ori">align</span>
  <span class="vitem item6">Text</span>
</div>

<div>
  <strong>text-bottom : </strong>
  <span class="ori">align</span>
  <span class="vitem item7">Text</span>
</div>

<div>
  <strong>middle : </strong>
  <span class="ori">align</span>
  <span class="vitem item8">Text</span>
</div>
```

```
.ori {
  font-size: 20px;
}

.vertical-item {
  font-size: 10px;
}

.item1 {
  vertical-align: baseline;
}

.item2 {
  vertical-align: sub;
}

.item3 {
  vertical-align: super;
}

.item4 {
  vertical-align: top;
}

.item5 {
  vertical-align: text-top;
}

.item6 {
  vertical-align: bottom;
}

.item7 {
  vertical-align: text-bottom;
}

.item8 {
  vertical-align: middle;
}
```

# 줄 바꿈 설정

white-space

```
p.item1 {  
  white-space: nowrap;  
}  
p.item2 {  
  white-space: pre;  
}  
p.item3 {  
  white-space: pre-line;  
}  
p.item4 {  
  white-space: pre-wrap;  
}
```

줄 바꿈이 없습니다

줄 바꿈, 띄어쓰기, 공백등 모든것이 보여지며,  
박스를 벗어나도 줄 바꿈이 일어나지 않습니다

줄 바꿈만 보여주며, 띄어쓰기는 무시합니다  
자동으로 줄바꿈이 됩니다

줄 바꿈과 띄어쓰기를 모두 보여주며  
자동으로 줄바꿈이 됩니다

# CSS 배경 스타일

CSS Background

CSS로 요소(Element)에 배경을 지정합니다

# 배경색

background-color

```
div {  
  background-color: #eee;  
  background-color: #efefef;  
  background-color: rgb(230, 222, 120);  
  background-color: rgba(230, 22, 120, 0.4);  
}
```

HEX Code (#RRGGBB or #RGB)

rgb(0~255값으로 표현)

alpha(투명도)를 조절



# 배경 이미지

background-image

```
div {  
    background-image: url('../images/icon.png');  
}
```

이미지의 주소는 상대경로, 절대경로 모두 사용가능합니다.

# 배경 이미지 반복

background-repeat

```
div {
```

```
background-repeat: no-repeat;
```

반복하지 않음

```
background-repeat: repeat;
```

가로세로 바둑판 형식으로 반복

```
background-repeat: repeat-x;
```

가로(x축)으로만 반복

```
background-repeat: repeat-y;
```

세로(y축)으로만 반복

```
}
```

가로로 반복하는 이미지의 경우, 세로로 길고 가로 1px인 이미지를 이용해 배경을 나타낼 수 있으며, 세로반복의 경우 반대로 가로로 길고 세로가 1px인 이미지를 이용해 배경을 나타낼 수 있습니다.

# 배경 이미지 반복

background-repeat

세로로 긴 이미지를  
repeat-x를 이용해  
가로로 반복하면  
배경화면이 됩니다  
(그라데이션 효과)



# 배경 이미지 위치

background-position

```
div {  
    background-position: 50% 16px;  
    background-position: center bottom;  
}
```

center는 가운데 (x,y모두 가능)  
left, right는 x축  
top, bottom은 y축에만 사용

삽입된 이미지의 좌표를 정해줍니다.

두 개의 값을 받으며, 각각 x축, y축의 값을 가집니다.

각각은 양의값을 가질경우 x축은 우측, y축은 하단으로 이동합니다. (음수를 가질경우 반대로 좌측, 상단으로 이동합니다)

left, center, right, top, bottom으로 x, y축의 0%, 100%, 가운데 값을 사용할 수 있습니다.

# 배경 이미지 고정

background-attachment

```
div {
```

```
background-attachment: local;
```

```
background-attachment: scroll;
```

```
background-attachment: fixed;
```

```
}
```

요소 안 내용에 고정  
(스크롤할 때 이미지가 같이 움직임)

요소에 고정 (스크롤에 무관)

background-position좌표를 웹 페이지 화면 전체기준으로 합니다

local값을 가질 경우, 요소의 왼쪽 상단을 기준으로 이미지를 표현하며, 스크롤 시 이미지가 같이 움직입니다.

scroll값을 가질 경우, 요소의 왼쪽 상단을 기준으로 이미지를 표현하며, 스크롤 시 이미지가 함께 움직이지 않고 고정됩니다.

fixed값을 가질 경우, 요소와 관계없이 웹 페이지 전체 화면을 기준으로 이미지가 표시되며, 스크롤 시 함께 움직이지 않고 고정됩니다.

# 배경 속기법

background

```
div {  
  background: url('images/sample01.png') no-repeat scroll right 50% #eee;  
}
```

image

repeat

attachment

position

color

지금까지 배운 background관련 속성을 한 번에 줄여쓸 수 있습니다.  
순서대로 image, repeat, attachment, position(x,y), color값을 나타냅니다.

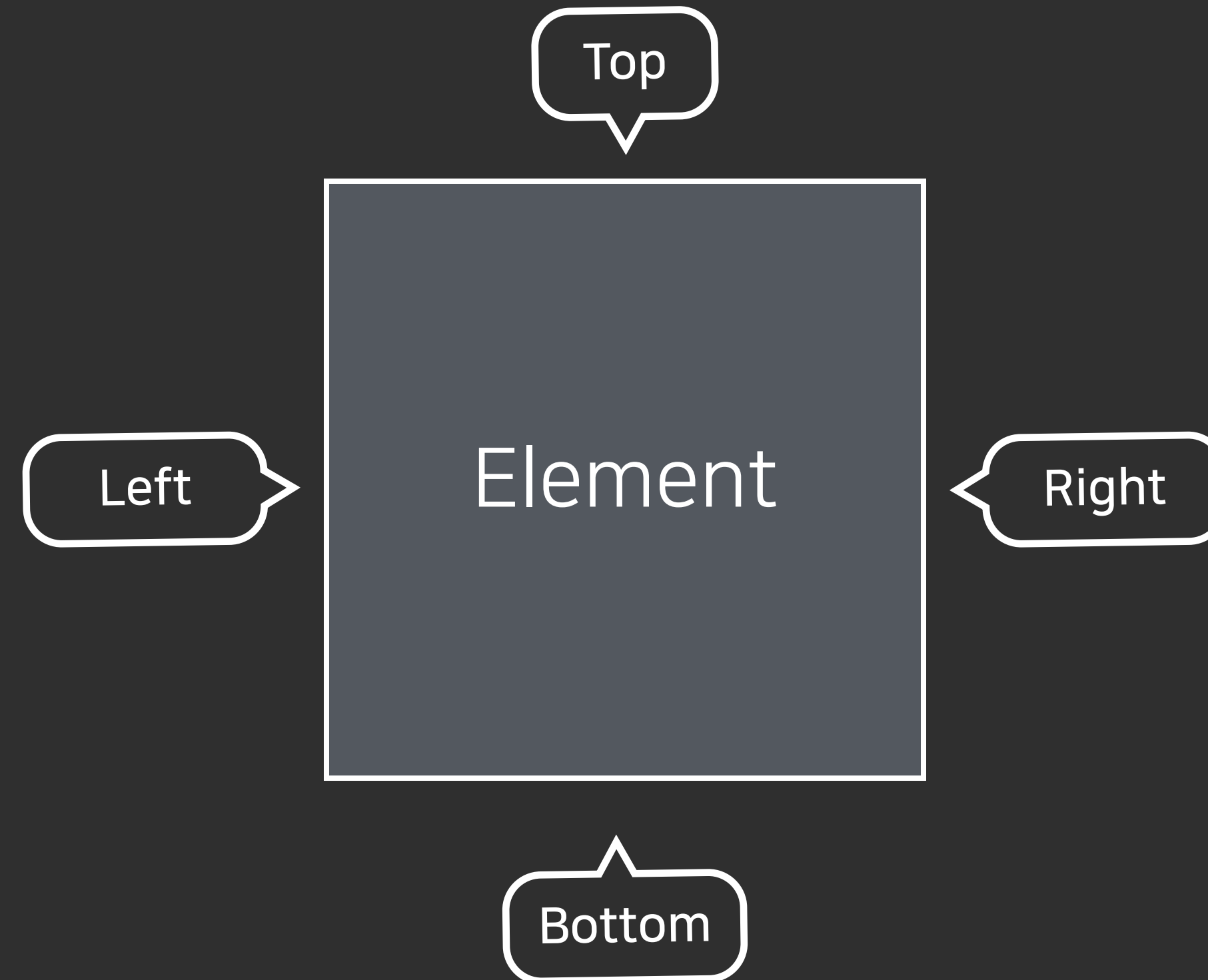
# CSS 테두리 스타일

CSS Border

CSS로 요소(Element)에 테두리를 지정합니다

# 요소의 방향

Element's direction



요소의 상하좌우 속성을 정의할 때, 순서는 시계방향으로 진행됩니다.

상단부터 시계방향으로, Top -> Right -> Bottom -> Left방향으로 값을 정한다고 기억하시면 됩니다.



# 테두리를 구성하는 요소

선 굵기 (border-width)

```
div {
```

```
border-width: 3px;
```

```
border-top-width: 4px;
```

```
border-width: 3px 4px 5px 6px;
```

```
border-width: 5px 10px;
```

```
}
```

상하좌우 모두 3px

상단만 4px

상 우 하 좌 순서대로 3,4,5,6px

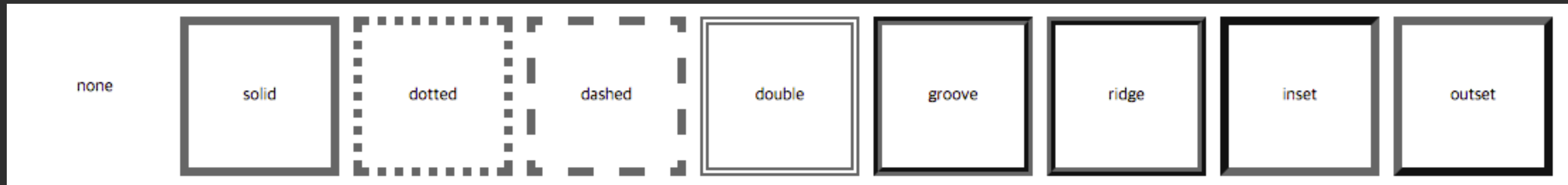
상하 5px, 좌우 10px

값을 2개만 적을 경우, 첫 번째 값은 상&하의 값, 두 번째 값은 좌&우의 값을 나타냅니다.

# 테두리를 구성하는 요소

선 형태 (border-style)

```
div {  
  border-style: solid;  
  border-top-style: double;  
  border-style: solid double dashed dotted;  
}
```



solid 실선

dotted 점선

dashed 바느질선 형태의 점선

double 이중선

groove 입체적으로 보여줌

ridge groove와 반대방향으로 선이 돌출

inset 요소 전체가 안으로 들어가 보임

outset 요소 전체가 바깥으로 나와 보임

# 테두리를 구성하는 요소

선 색상 (border-color)

```
div {  
    border-color: #aaa;  
    border-color: red blue green yellow;  
}
```

지금까지와 마찬가지로 Hex code, rgb, rgba, ColorName 전부 사용 가능합니다.

# 테두리 속성 속기법

border

```
div {  
    border: 1px solid red;  
}
```

border의 속기법은 모든 변에 동일한 값만 적용 가능합니다.

(각 변에 다른 값을 주고 싶을 경우, 각 속성(width, style등)에 4가지 값을 입력하거나, border-top-<property>에 값을 입력해야 합니다.

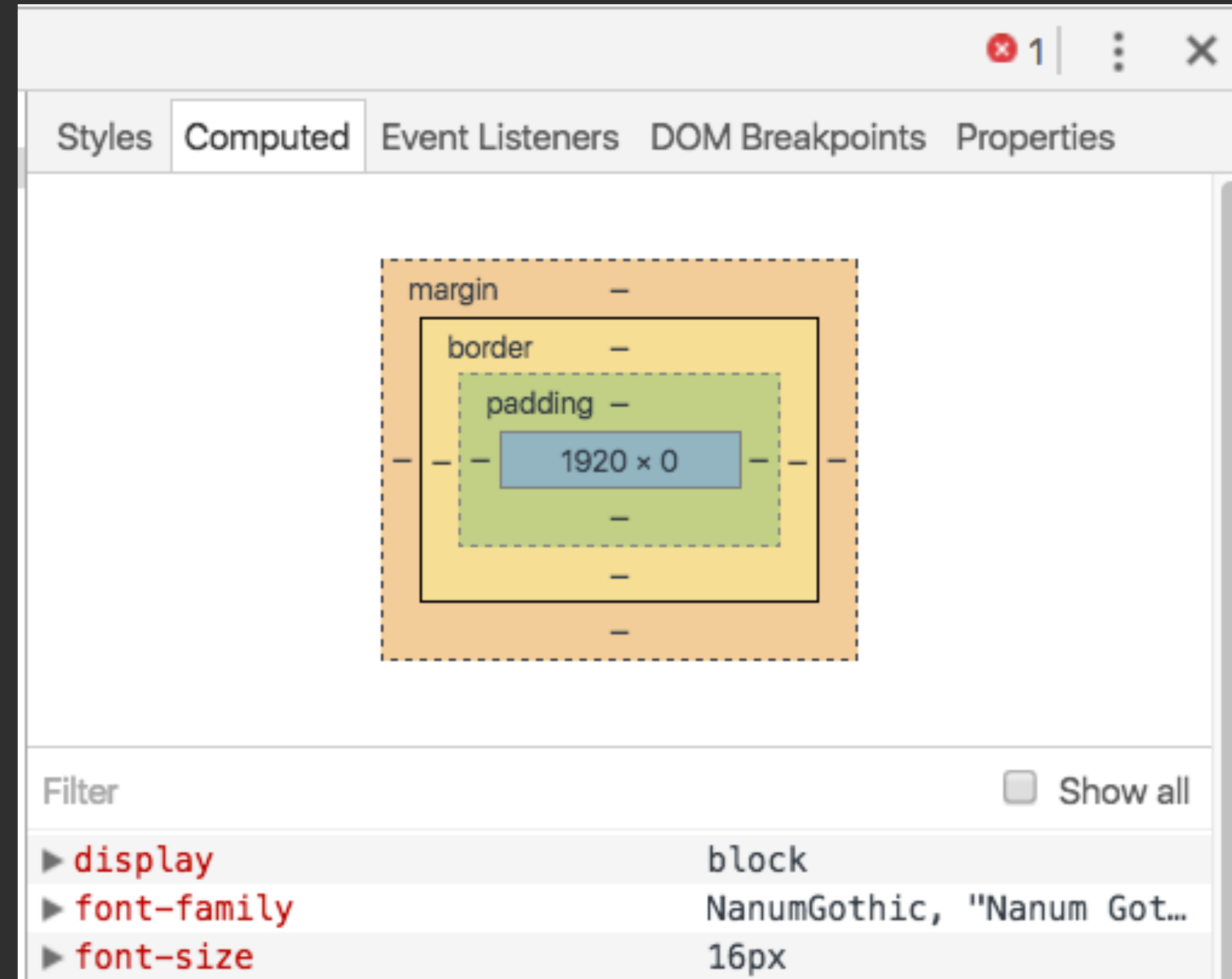
# CSS 박스 모델

CSS Box Model

CSS로 요소를 구성하는 박스에 대해 알아봅니다

# 박스모델?

CSS요소를 이루는 형태

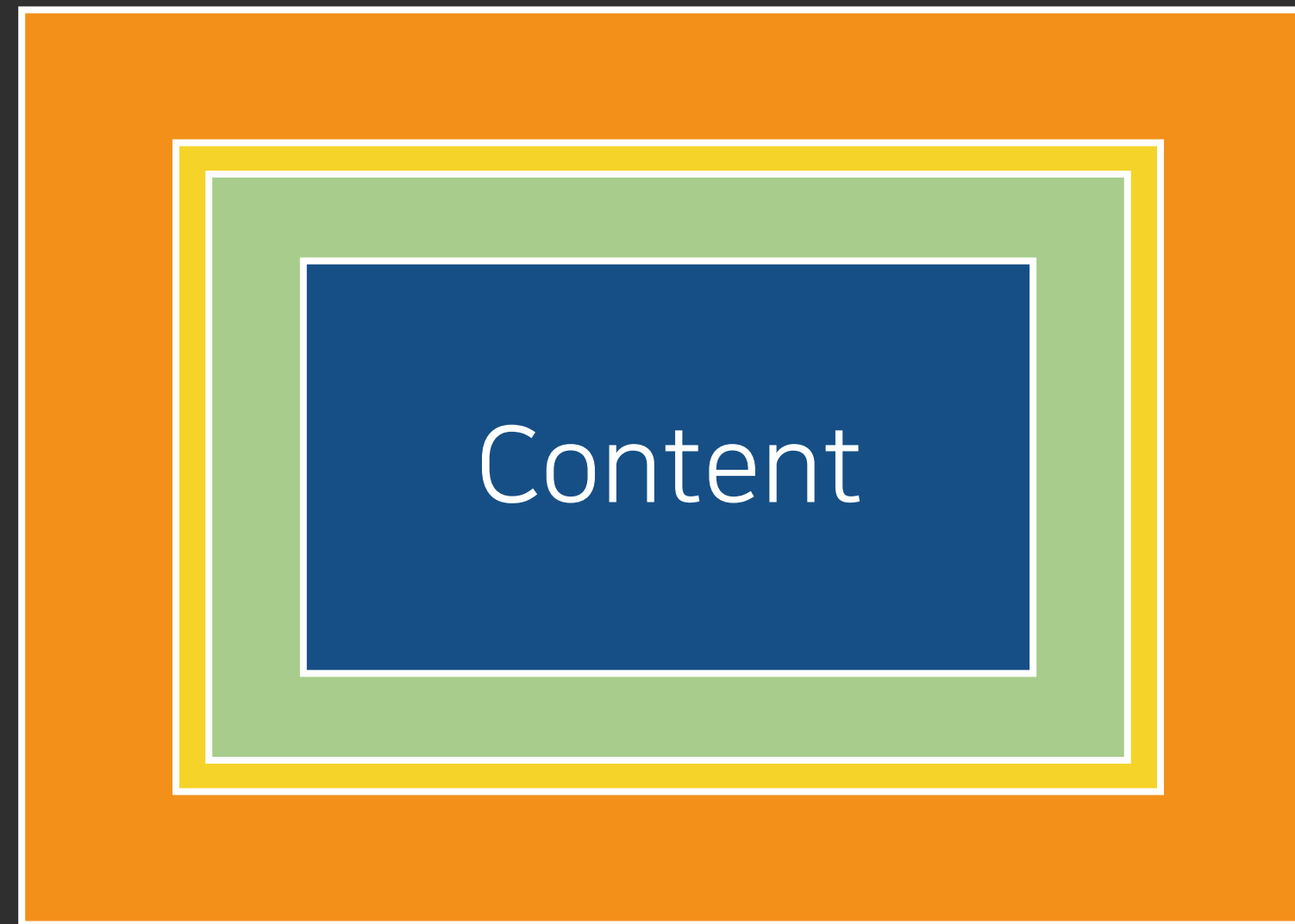


개발자 도구에서 CSS요소의 박스 모델을 확인 가능

CSS요소는 박스 형태를 이루며,  
박스는 콘텐츠, 패딩, 보더, 마진의 4가지로 이루어집니다

# 박스모델?

CSS요소를 이루는 형태



CSS요소는 박스 형태를 이루며,  
박스는 콘텐츠, 패딩, 보더, 마진의 4가지로 이루어집니다

# 블록 요소와 인라인 요소의 차이

인라인 요소는 가로마진만 가질 수 있습니다

Inline Element

Block Element

블록 요소는 가로/세로 마진을 모두 가집니다

인라인 요소의 길이/높이는 지정이 불가능합니다 (내용에 자동으로 맞추어짐)



# 바깥 여백 (마진)

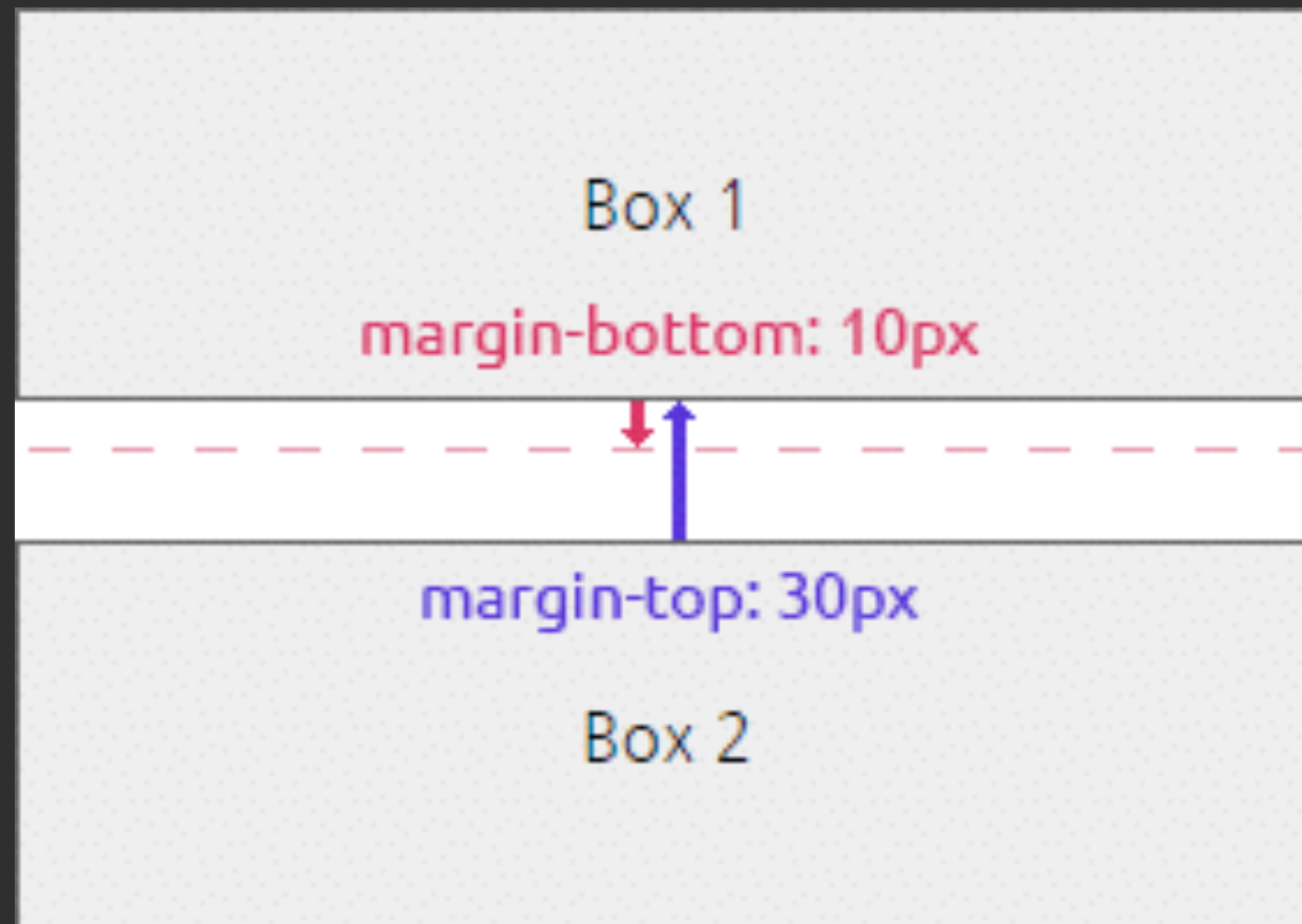
margin

```
div {  
    margin-top: 10px;  
    margin-bottom: 30px;  
    margin: 10px 0;  
    margin: 40px 20px 30px 50px;  
}
```

# 마진 겹침

margin collapse

```
<div style="margin-bottom:10px;">Box 1</div>  
<div style="margin-top:30px;">Box 2</div>
```



두 블록요소의 마진이 서로 겹칠 경우, 해당하는 마진값이 더해지는 것이 아니라 둘 중 큰 값만이 적용됩니다.

(세로가 아닌 가로에서는 해당 현상이 없습니다)

따라서 서로 위/아래로 겹치는 마진값을 준 경우, 한 쪽에만 값을 몰아주거나, padding을 활용하는 방식으로 해결해야 합니다.

# 내부 여백 (패딩)

padding

```
div {  
    padding-top: 10px;  
    padding-bottom: 10px;  
    padding: 10px 0;  
    padding: 10px 20px 30px 40px;  
}
```

padding과 margin을 구분하는 가장 쉬운법은,

padding과 margin을 준 요소에 background-color를 지정한 후 개발자 모드에서 해당 요소가 차지하는 공간을 확인하는 것입니다.

# 가로, 세로

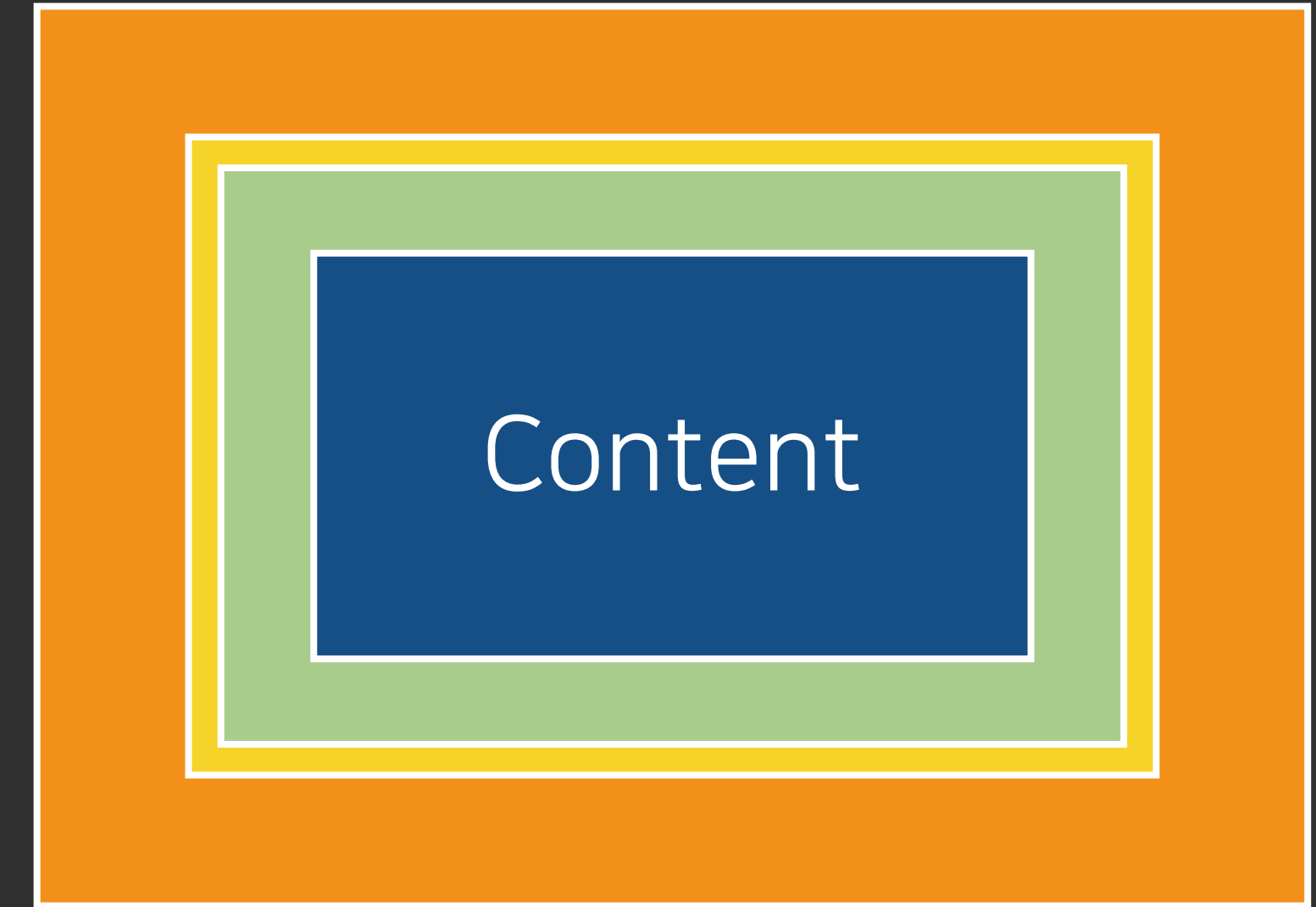
width, height

```
div {  
  width: 100px;  
  height: 50px;  
}
```

# 가로, 세로

width, height

```
div {  
  width: 200px;  
  height: 50px;  
  padding: 10px;  
  border: 1px solid black;  
  margin: 15px;  
}
```



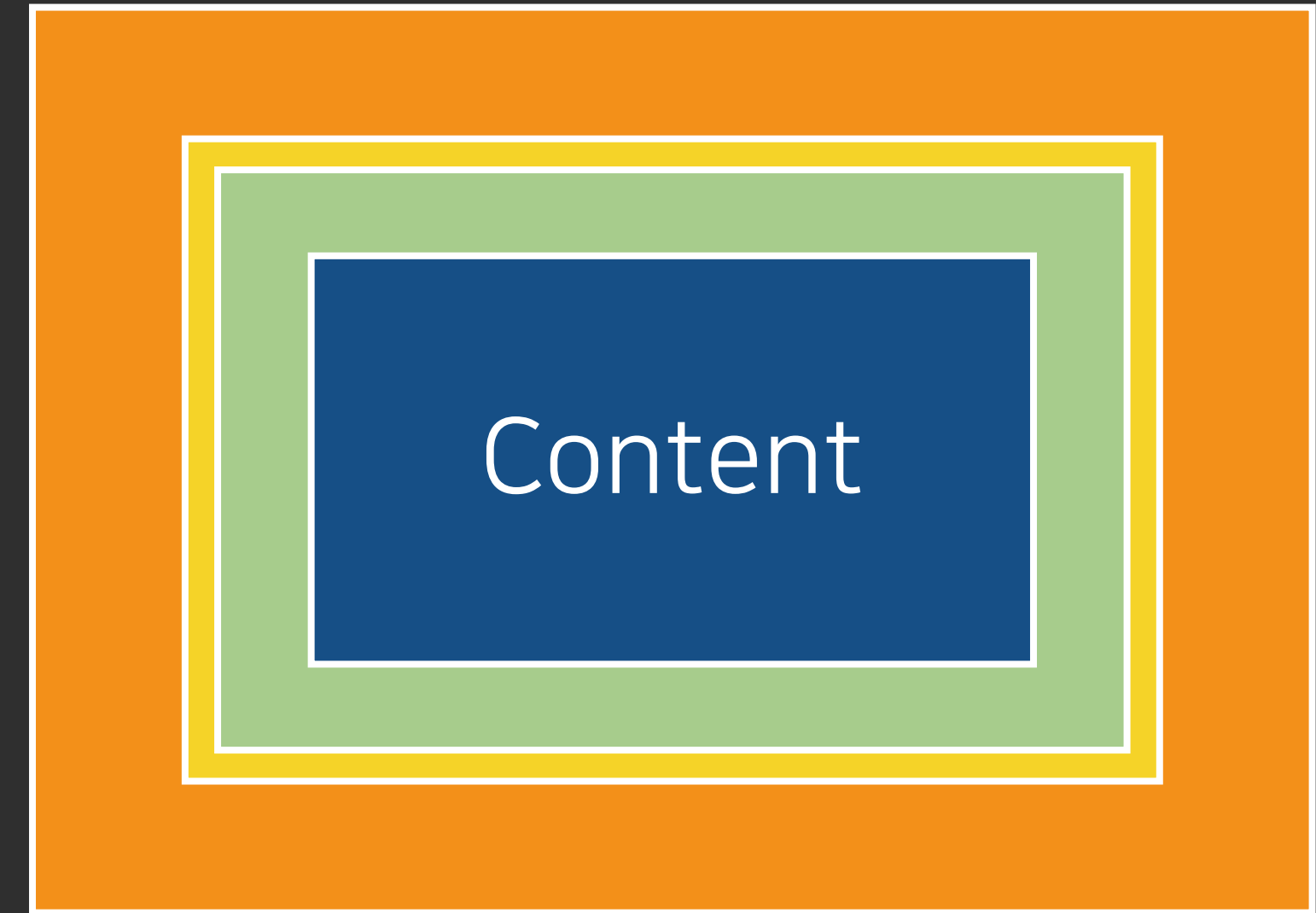
요소의 총 가로길이  
 $200\text{px}(\text{가로}) + (10\text{px} + 1\text{px} + 15\text{px}) \times 2$   
 $= 252\text{px}$

padding과 margin을 구분하는 가장 쉬운법은,  
padding과 margin을 준 요소에 background-color를 지정한 후 개발자 모드에서 해당 요소가 차지하는 공간을 확인하는 것입니다.

# box-sizing

박스 모델의 크기를 지정합니다

```
div {  
  width: 200px;  
  height: 50px;  
  padding: 10px;  
  border: 1px solid black;  
  margin: 15px;  
  box-sizing: border-box;  
}
```



요소의 총 가로길이 200px 고정 (margin제외)  
Content의 가로길이는  $200\text{px} - (10\text{px} + 1\text{px}) \times 2 = 178\text{px}$

box-sizing에 border-box를 지정하면, 요소의 width값에 맞추어 padding, border값을 제외한 width가 새로 적용됩니다.

# CSS 리스트 스타일

CSS List Style

CSS로 리스트에 스타일을 지정합니다

# 리스트 앞 Bullet타입 설정

list-style-type

```
ul {  
  list-style-type: disc;  
  list-style-type: circle;  
  list-style-type: square;  
  
  list-style-type: decimal;  
  list-style-type: lower-alpha;  
  list-style-type: upper-alpha;  
  list-style-type: lower-roman;  
  list-style-type: upper-roman;  
}
```

disc, circle, square는 Unordered List에 어울리는 속성이며,  
decimal, alpha, roman은 Ordered List에 어울리는 속성입니다.

## 적용 예시

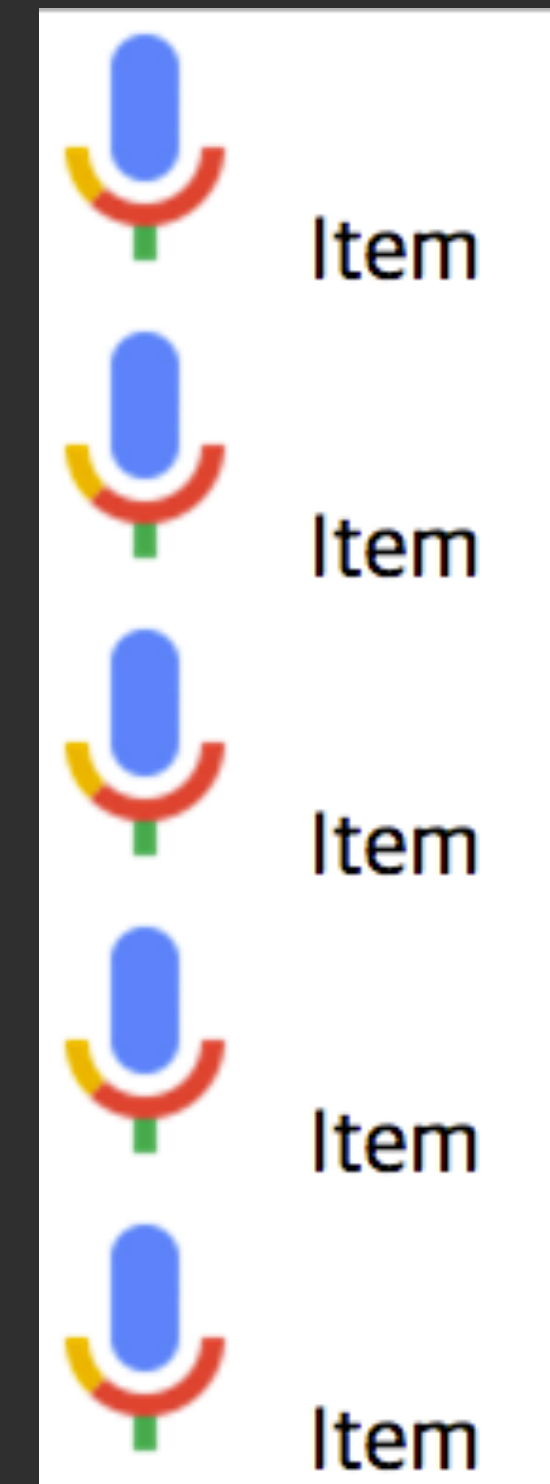
- disc 형태
  - circle 형태
  - square 형태
- 
1. decimal
  - b. lower-alpha
  - C. upper-alpha
  - iv. lower-roman
  - V. upper-roman



# 리스트 블릿에 이미지 지정

list-style-image

```
ul {  
  list-style-image: url('images/mic.png');  
}
```



padding과 margin을 구분하는 가장 쉬운법은,  
padding과 margin을 준 요소에 background-color를 지정한 후 개발자 모드에서 해당 요소가 차지하는 공간을 확인하는 것입니다.

# 리스트 블릿 위치 지정

list-style-position

```
ul {  
  list-style-position: inside;  
  list-style-position: outside;  
}
```

## 적용 예시

- outside는 기본적인 리스트 형태입니다.  
이렇게 줄 바꿈이 일어나도 블릿과 일정한 여백을 유지합니다.
- 반면 inside는 본문 내에 위치하게 됩니다.  
때문에 이렇게 줄 바꿈이 일어나면 마치 본문의 일부처럼 보일 수 있습니다.

참고로 위의 리스트의 여백은 ul 요소의 padding으로 조절할 수 있습니다.

# 리스트 스타일 속기법

list-style

```
ul {
```

```
list-style: square url('images/mic.png') inside;
```

```
}
```

블릿 타입

이미지를 사용할 경우

블릿 위치 내부/외부 여부

\*\*블릿타입과 이미지 주소를 동시에 넣을 경우, 이미지를 찾지 못하면 지정한 블릿타입을 사용합니다.

# CSS의 테이블 스타일

CSS Table style

CSS로 테이블을 테이블에 스타일을 지정합니다

# 테두리 합치기

border-collapse

```
<table>
  <tr>
    <td>A</td>
    <td>B</td>
    <td>C</td>
  </tr>
  <tr>
    <td>1</td>
    <td>2</td>
    <td>3</td>
  </tr>
  <tr>
    <td>4</td>
    <td>5</td>
    <td>6</td>
  </tr>
</table>
```

```
tr, td {
  border: 1px solid black;
  width: 30px;
  text-align: center;
}
```

A	B	C
1	2	3
4	5	6

tr, td에 테두리를 준 경우

# 테두리 합치기

border-collapse

```
<table>
  <tr>
    <td>A</td>
    <td>B</td>
    <td>C</td>
  </tr>
  <tr>
    <td>1</td>
    <td>2</td>
    <td>3</td>
  </tr>
  <tr>
    <td>4</td>
    <td>5</td>
    <td>6</td>
  </tr>
</table>
```

```
table {
  border-collapse: collapse;
}
tr, td {
  border: 1px solid black;
  width: 30px;
  text-align: center;
}
```

A	B	C
1	2	3
4	5	6

border-collapse 적용

테이블의 셀간 공백을 합쳐서 없애는 속성입니다.  
해당속성은 오직 table요소에만 사용 가능합니다.

# 테이블 셀 간격

border-spacing

```
<table>
  <tr>
    <td>A</td>
    <td>B</td>
    <td>C</td>
  </tr>
  <tr>
    <td>1</td>
    <td>2</td>
    <td>3</td>
  </tr>
  <tr>
    <td>4</td>
    <td>5</td>
    <td>6</td>
  </tr>
</table>
```

```
table {
  border-spacing: 10px;
}
tr, td {
  border: 1px solid black;
  width: 30px;
  text-align: center;
}
```

A	B	C
1	2	3
4	5	6

border-spacing 적용

\*\*셀 간 간격을 지정할 때는, border-collapse가 'collapse'값이면 적용되지 않습니다.

# 빈 셀의 표시

empty-cells

```
<table>
  <tr>
    <td>A</td>
    <td>B</td>
    <td>C</td>
  </tr>
  <tr>
    <td>1</td>
    <td></td>
    <td></td>
  </tr>
  <tr>
    <td>4</td>
    <td>5</td>
    <td></td>
  </tr>
</table>
```

```
table {
  border-spacing: 10px;
}
tr, td {
  border: 1px solid black;
  width: 30px;
  text-align: center;
  empty-cells: hide;
}
```

A	B	C
1		
4	5	

empty-cells에 show 적용

A	B	C
1		
4	5	

empty-cells에 hide 적용

\*\*셀 간 간격을 지정할 때는, border-collapse가 'collapse'값이면 적용되지 않습니다.



# 테이블 레이아웃

table-layout

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laudantium, neque.	Lorem.
--	--------

테이블의 기본 설정은 내용이 긴 쪽이 더 늘어납니다  
table-layout: auto;

# 테이블 레이아웃

table-layout

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Laudantium, neque.	Lorem.
--	--------

table-layout의 속성을 fixed로 설정하면, 셀 길이가 일정하게 유지됩니다.  
(이 때, table에는 width property가 설정되어있어야 합니다)

배운내용을 복습하며 문서를 작성해봅시다  
<https://namu.wiki/w/박보영>

# CSS의 화면 표시 속성

CSS display

CSS로 요소가 어떤 모습으로 보일지 결정합니다

# 화면 표시방법

display: block

span은 원래 인라인속성이지만,  
display 프로퍼티에 block값을 받으면 블록 속성을 지니게 됩니다

```
span {  
  display: block;  
}
```

기존에 블록 요소가 아닌 요소를 블록 속성을 갖도록 합니다.

# 화면 표시방법

display: inline

반대로, div가 display프로퍼티에 inline속성을 받으면  
인라인 요소로 취급됩니다

```
div {  
  display: inline;  
}
```

기존에 인라인 요소가 아닌 요소를 인라인 속성을 갖도록 합니다.

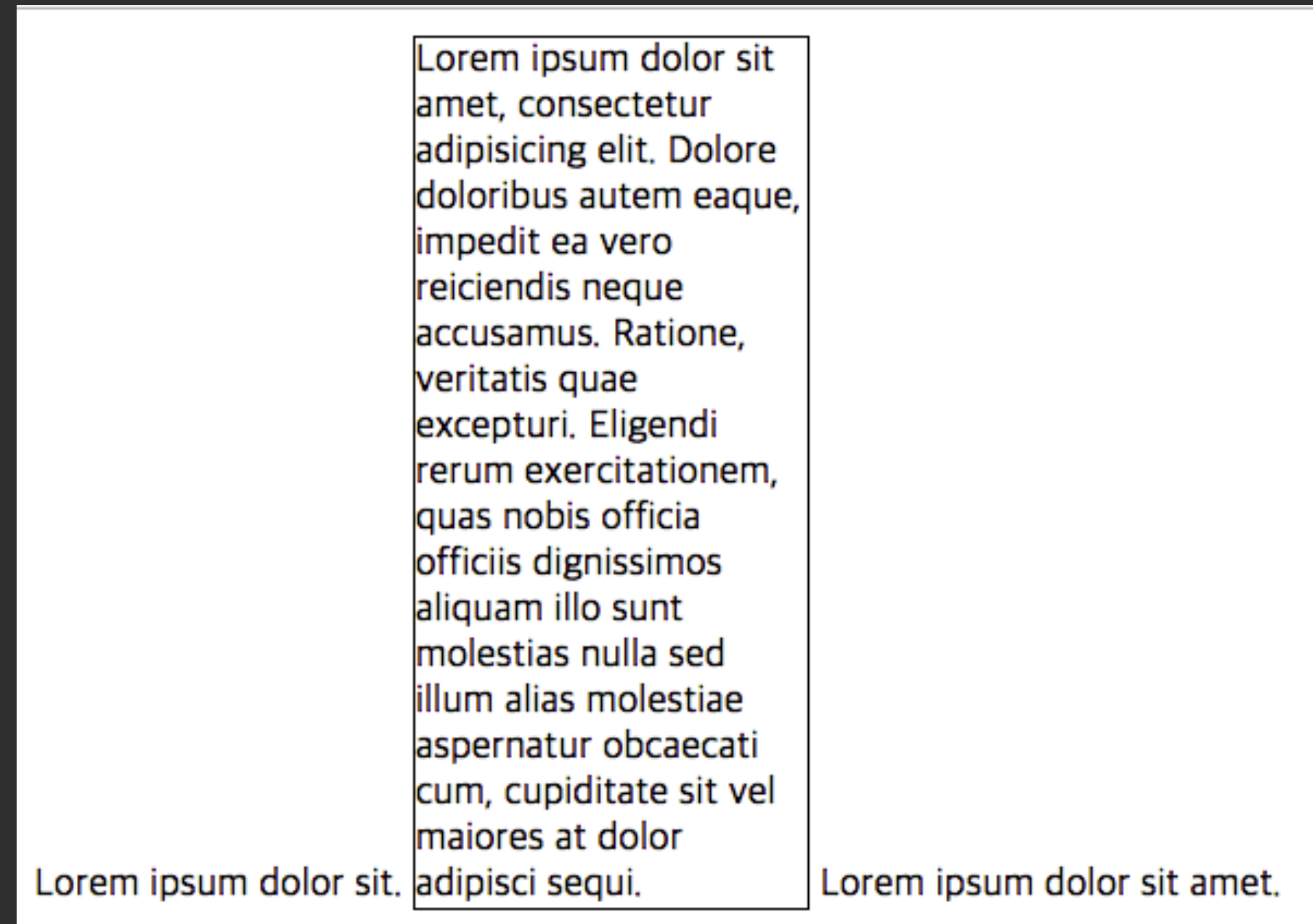
# 화면 표시방법

display: inline-block

기본적으로 inline요소처럼 취급되지만,  
block요소와 같이 높이 및 상/하값을 가질 수 있습니다

```
span {  
  display: inline-block;  
}
```

인라인요소에 높이와 상/하 패딩, 마진값을 줄 때 사용합니다.



display 프로퍼티에 inline-block속성을 가진 span

# 화면 표시방법

display: none

화면에 완전히 보이지 않게 됩니다

```
div {  
  display: none;  
}
```

해당 요소의 하위 요소들도 전부 보이지 않게 되며, 공간도 차지하지 않습니다.



# 화면 표시방법

visibility

화면에 공간은 차지하나, 투명해집니다

```
div {  
  visibility: hidden;  
  visibility: visible;  
}
```

display: none;은 화면에서 차지하던 공간조차도 전부 없어지며,  
visibility: hidden;은 화면에서 차지하던 공간은 그대로인채 투명해진다고 생각하시면 됩니다.

# 화면 넘침 표시방법

overflow

```
div {  
  overflow: hidden;  
  overflow: visible;  
  overflow: auto;  
  overflow: scroll;  
}
```

넘친 콘텐츠를 전부 숨깁니다

영역 밖으로 콘텐츠가 나간 모습이 보입니다

콘텐츠가 넘칠 경우, 스크롤바가 생성됩니다

콘텐츠가 넘치지 않아도 항상 스크롤바가 있습니다

**visible**

이 속성 값을 사용한 요소는 안에 콘텐츠들이 박스를 벗어나더라도 그대로

**scroll**

스크롤이 항상

**hidden**

당신은 이 박스를 벗어나는 내용을 볼 수 없습니다. 왜냐

**auto**

내용이 짧으면 스크롤 안보임.

**auto**

하지만 내부의 콘텐츠가 박스를 벗어나기 시작하면

높이가 고정되어있고, 내용이 길면 스크롤 할 부분에서는 auto를 사용합니다.

# CSS float속성

CSS float

CSS로 요소를 띄웁니다

# float속성

float: 띄우다



## 피카츄

게임 및 애니메이션 포켓몬스터 시리즈에 등장하는 바와 같이 귀요미. 그 때문인지 포켓몬 약 700마리 이상에 달하는 포켓몬스터의 카운터파트인 그나마 피카츄가 가장 귀여워서 인기가 많았

float속성은 중간에 이미지를 넣은 단락을 만들고자 하는 경우에도 사용가능합니다.

# float속성

float: 띄우다

float: right; 적용시

Element1

Element2

Element3



Element2

Element3

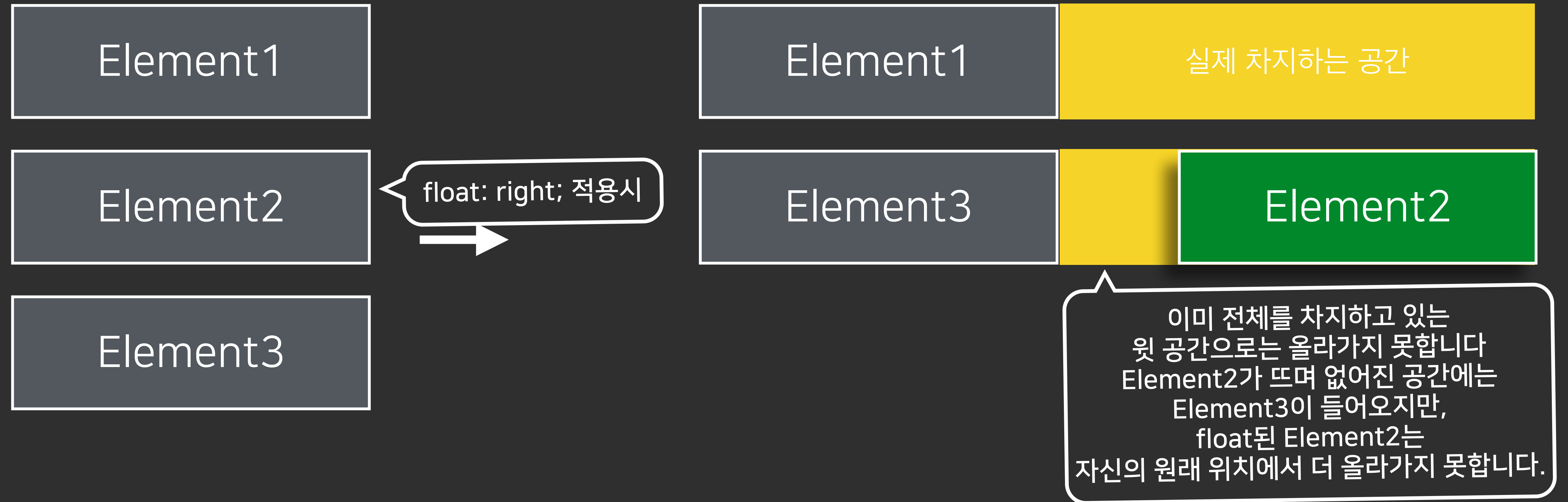
Element1

기존 항목들보다 위쪽에 떠서  
Element2가 빈 공간으로 올라옵니다

float를 사용하면 해당 요소를 문서의 흐름과 별개로 취급하여, 왼쪽이나 오른쪽으로 띄워줄 수 있습니다.

# float속성

float: 띄우다



Element1이 block요소라면, 해당 요소가 가로 너비를 차지하므로 그 아래쪽에서 우측으로 가게됩니다

# float속성

float: 띄우다

전체에 float:right 적용시

Element1

Element2

Element3



Element3

Element2

Element1

모든 요소가 right로 뜨며 block속성을 잃고  
차례대로 배치됩니다

# float속성

float: 띄우다

float:left

float:left

Element1

Element2

Element3

float:right



Element1

Element2

왼쪽부터 left속성끼리 차례로 정렬되며

우측부터 right속성이 쌓입니다

Element3



# float속성

float: 띄우다

전체에 float:left 적용시

Element1

Element2

Element3



부모 엘리먼트보다 가로가 길 경우, 아래로 내려갑니다

Element1

Element2

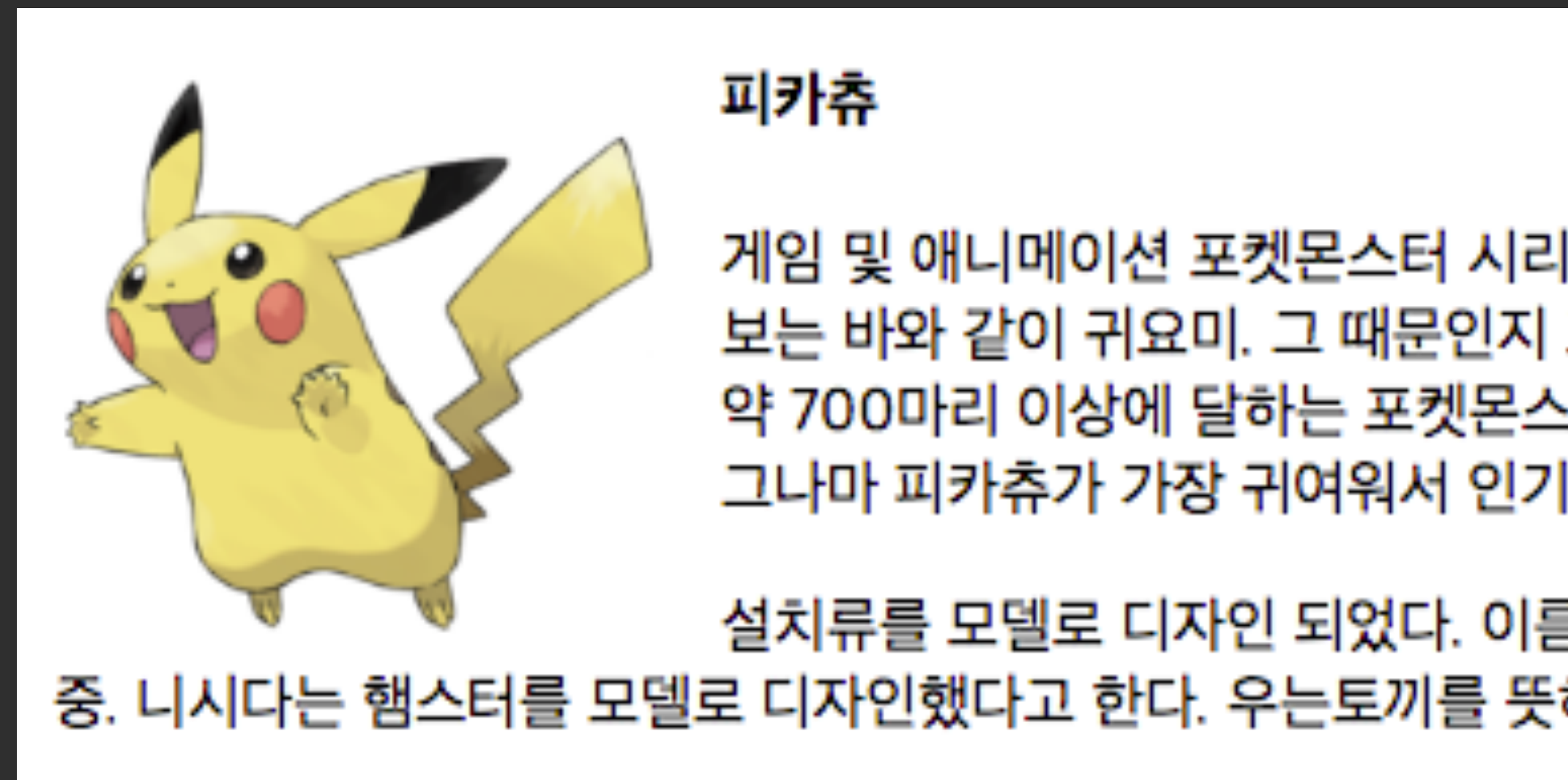
Element3

Element1이 block요소라면, 해당 요소가 가로 너비를 차지하므로 그 아래쪽에서 우측으로 가게됩니다

# clear 속성

float요소와 겹치는 경우, float속성을 해제합니다

float:left

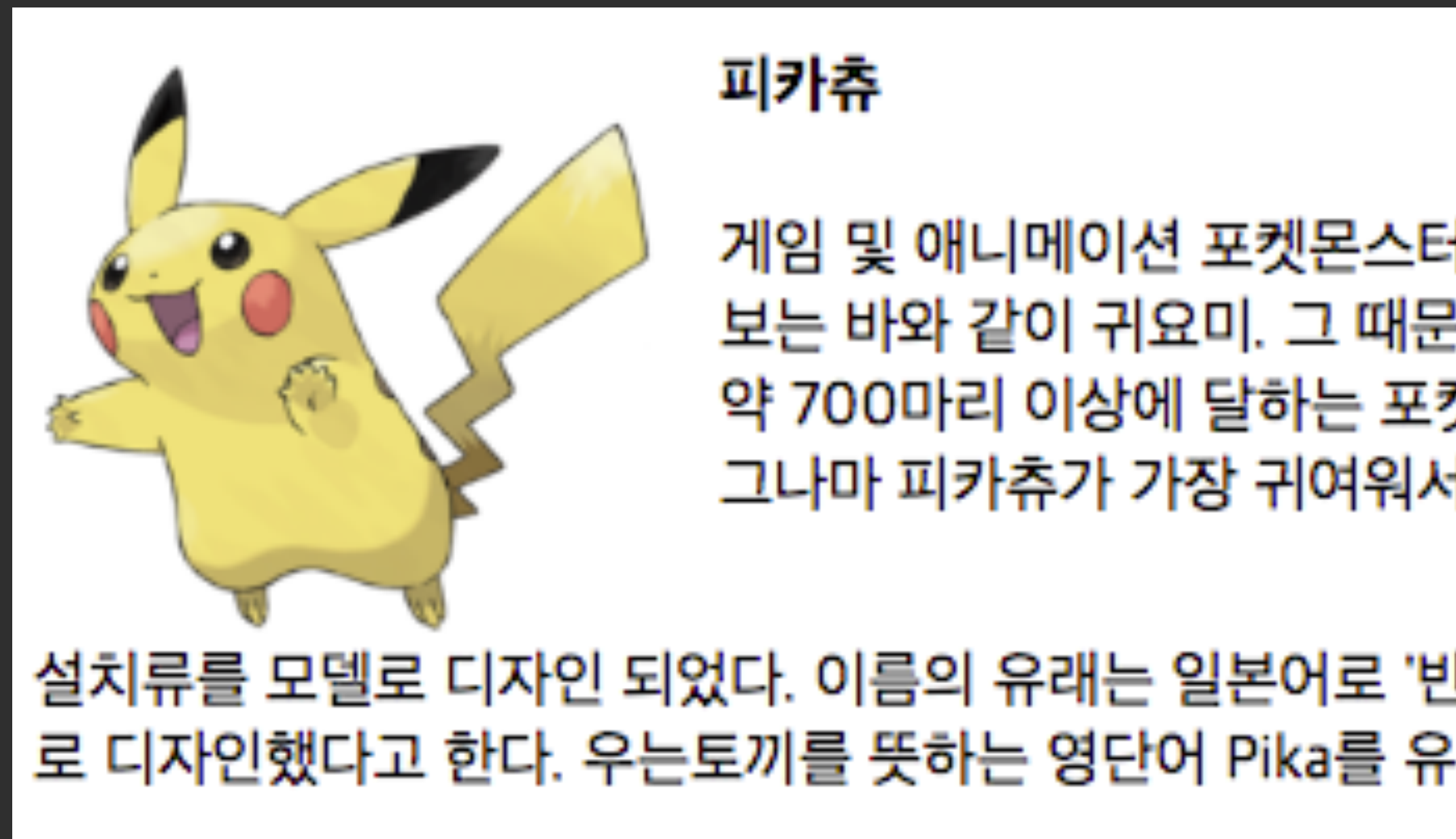


두 번째 문단은 이미지의 float에 관계없이 block처리되어 아래로 내려가도록 하려면?

# clear 속성

float요소와 겹치는 경우, float속성을 해제합니다

float:left



아래쪽 요소에 {clear: left;} 지정

# clear 속성

float요소와 겹치는 경우, float속성을 해제합니다

```
p {  
  clear: both;  
  clear: left;  
  clear: right;  
}
```

left, right모두 해제

# CSS float 레이아웃

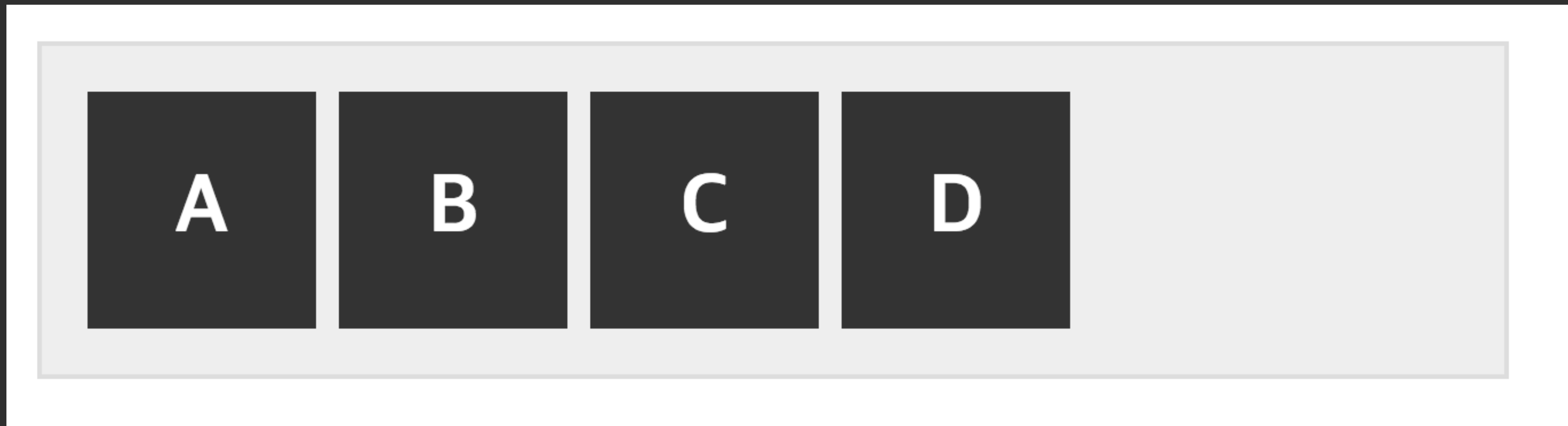
CSS float layout

CSS의 float속성으로 레이아웃을 구현합니다

# float 레이아웃

float layout

지금까지는 없었지만, 앨범이나 슬라이드 등에서 자주 쓰이는 레이아웃입니다



CSS로 레이아웃을 구현할 때는 float속성을 자주 사용합니다.

# float 레이아웃

float layout

CSS

```
.float-frame {  
  width: 300px;  
  background-color: #eee;  
  border: 1px solid #ddd;  
  padding: 10px;  
}  
  
.float-unit {  
  width: 50px;  
  background: #333;  
  color: #fff;  
  font-size: 18px;  
  font-weight: bold;  
  text-align: center;  
  padding: 15px 0;  
  margin-right: 5px;  
}
```

HTML

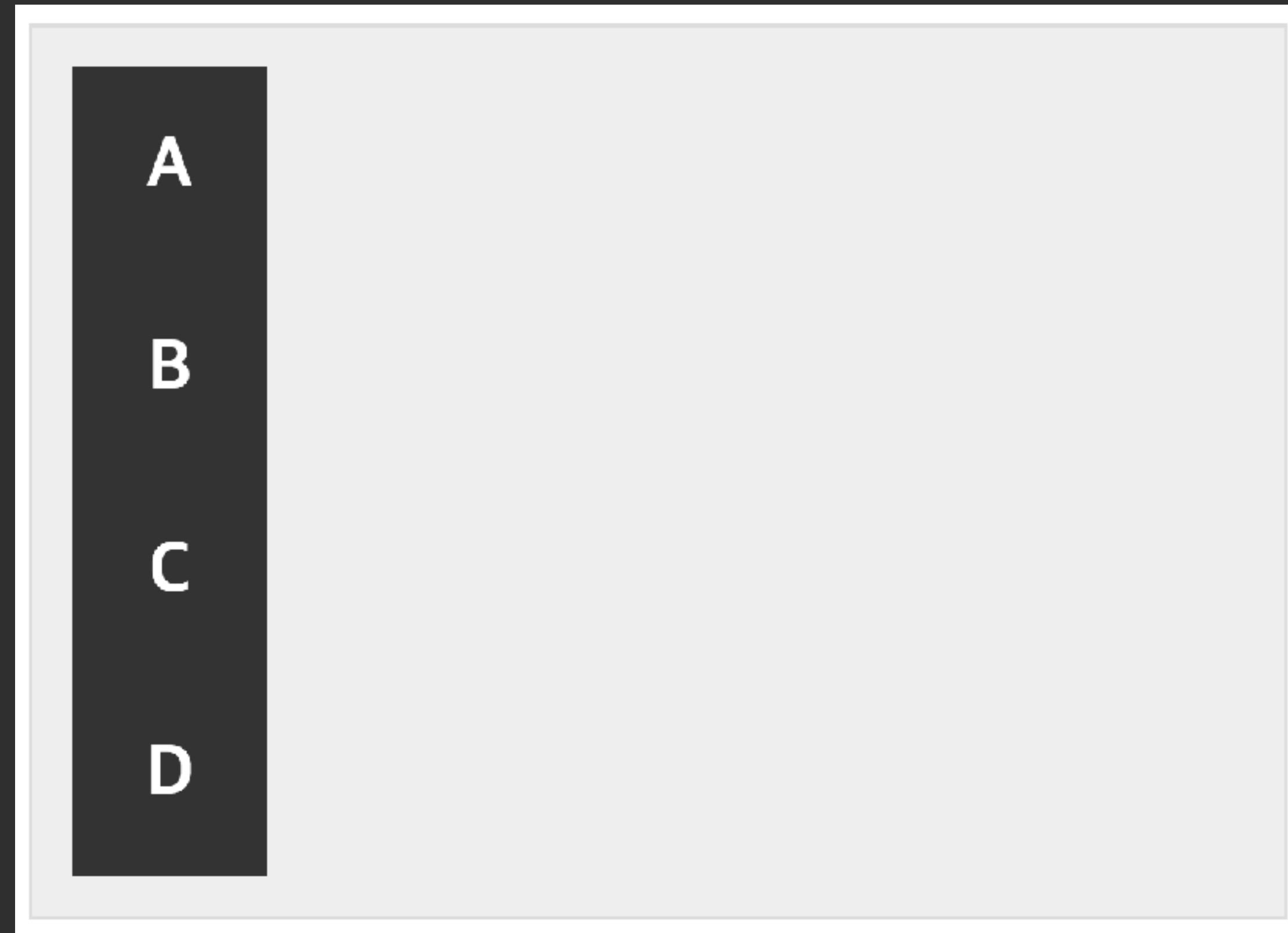
```
<div class="float-frame">  
  <div class="float-unit">A</div>  
  <div class="float-unit">B</div>  
  <div class="float-unit">C</div>  
  <div class="float-unit">D</div>  
</div>
```

CSS로 레이아웃을 구현할 때는 float속성을 자주 사용합니다.

# float 레이아웃

float layout

아직 float가 적용되지 않은 상태입니다



검은 span요소에 float를 적용합니다.



# float 레이아웃

float layout

## CSS

```
.float-frame {  
  width: 300px;  
  background-color: #eee;  
  border: 1px solid #ddd;  
  padding: 10px;  
}  
  
.float-unit {  
  width: 50px;  
  background: #333;  
  color: #fff;  
  font-size: 18px;  
  font-weight: bold;  
  text-align: center;  
  padding: 15px 0;  
  margin-right: 5px;  
  float: left;  
}
```

## HTML

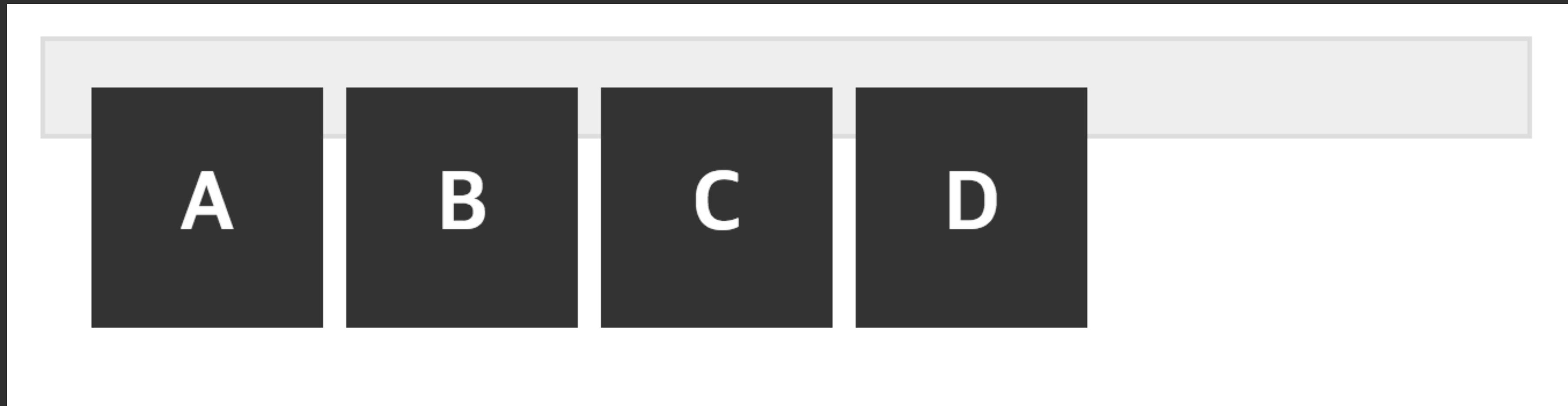
```
<div class="float-frame">  
  <div class="float-unit">A</div>  
  <div class="float-unit">B</div>  
  <div class="float-unit">C</div>  
  <div class="float-unit">D</div>  
</div>
```

float속성을 추가합니다

# float 레이아웃

float layout

항목들은 float:left;를 따라 잘 배열됐지만, 부모요소의 height가 인식되지 않습니다



# float 레이아웃 - 임의의 요소 삽입

float를 해제할 임의의 요소 삽입

CSS

```
.float-frame {  
  width: 300px;  
  background-color: #eee;  
  border: 1px solid #ddd;  
  padding: 10px;  
}  
  
.float-unit {  
  width: 50px;  
  background: #333;  
  color: #fff;  
  font-size: 18px;  
  font-weight: bold;  
  text-align: center;  
  padding: 15px 0;  
  margin-right: 5px;  
  float: left;  
}
```

HTML

```
<div class="float-frame">  
  <div class="float-unit">A</div>  
  <div class="float-unit">B</div>  
  <div class="float-unit">C</div>  
  <div class="float-unit">D</div>  
  <br style="clear: both;">  
</div>
```

# float 레이아웃 - overflow를 사용

부모에 overflow를 설정

CSS

```
.float-frame {  
  width: 300px;  
  background-color: #eee;  
  border: 1px solid #ddd;  
  padding: 10px;  
  overflow: auto;  
  overflow: hidden;  
}
```

```
.float-unit {  
  width: 50px;  
  background: #333;  
  color: #fff;  
  font-size: 18px;  
  font-weight: bold;  
  text-align: center;  
  padding: 15px 0;  
  margin-right: 5px;  
  float: left;  
}
```

HTML

```
<div class="float-frame">  
  <div class="float-unit">A</div>  
  <div class="float-unit">B</div>  
  <div class="float-unit">C</div>  
  <div class="float-unit">D</div>  
</div>
```

overflow:hidden은 overflow:visible때는 인식하지 못하던 float요소와, 자식요소의 margin값을 인식하게 됩니다.

# float 레이아웃 - 부모에도 float를 설정

부모에도 float를 설정

CSS

```
.float-frame {  
  width: 300px;  
  background-color: #eee;  
  border: 1px solid #ddd;  
  padding: 10px;  
  float: left;  
}  
  
.float-unit {  
  width: 50px;  
  background: #333;  
  color: #fff;  
  font-size: 18px;  
  font-weight: bold;  
  text-align: center;  
  padding: 15px 0;  
  margin-right: 5px;  
  float: left;  
}
```

HTML

```
<div class="float-frame">  
  <div class="float-unit">A</div>  
  <div class="float-unit">B</div>  
  <div class="float-unit">C</div>  
  <div class="float-unit">D</div>  
</div>
```

overflow:hidden은 overflow:visible때는 인식하지 못하던 float요소와, 자식요소의 margin값을 인식하게 됩니다.

# float 레이아웃 - 가상선택자를 이용

:after 가상선택자를 사용

## CSS

```
.float-frame {  
  width: 300px;  
  background-color: #eee;  
  border: 1px solid #ddd;  
  padding: 10px;  
}  
.float-frame::after {  
  content: " ";  
  display: block;  
  height: 0;  
  clear: both;  
}  
.float-unit {  
  width: 50px;  
  background: #333;  
  color: #fff;  
  font-size: 18px;  
  font-weight: bold;  
  text-align: center;  
  padding: 15px 0;  
  margin-right: 5px;  
  float: left;  
}
```

## HTML

```
<div class="float-frame">  
  <div class="float-unit">A</div>  
  <div class="float-unit">B</div>  
  <div class="float-unit">C</div>  
  <div class="float-unit">D</div>  
</div>
```

# CSS 포지션

CSS position

CSS의 요소의 위치를 설정합니다

# position

요소의 위치를 지정합니다





# position

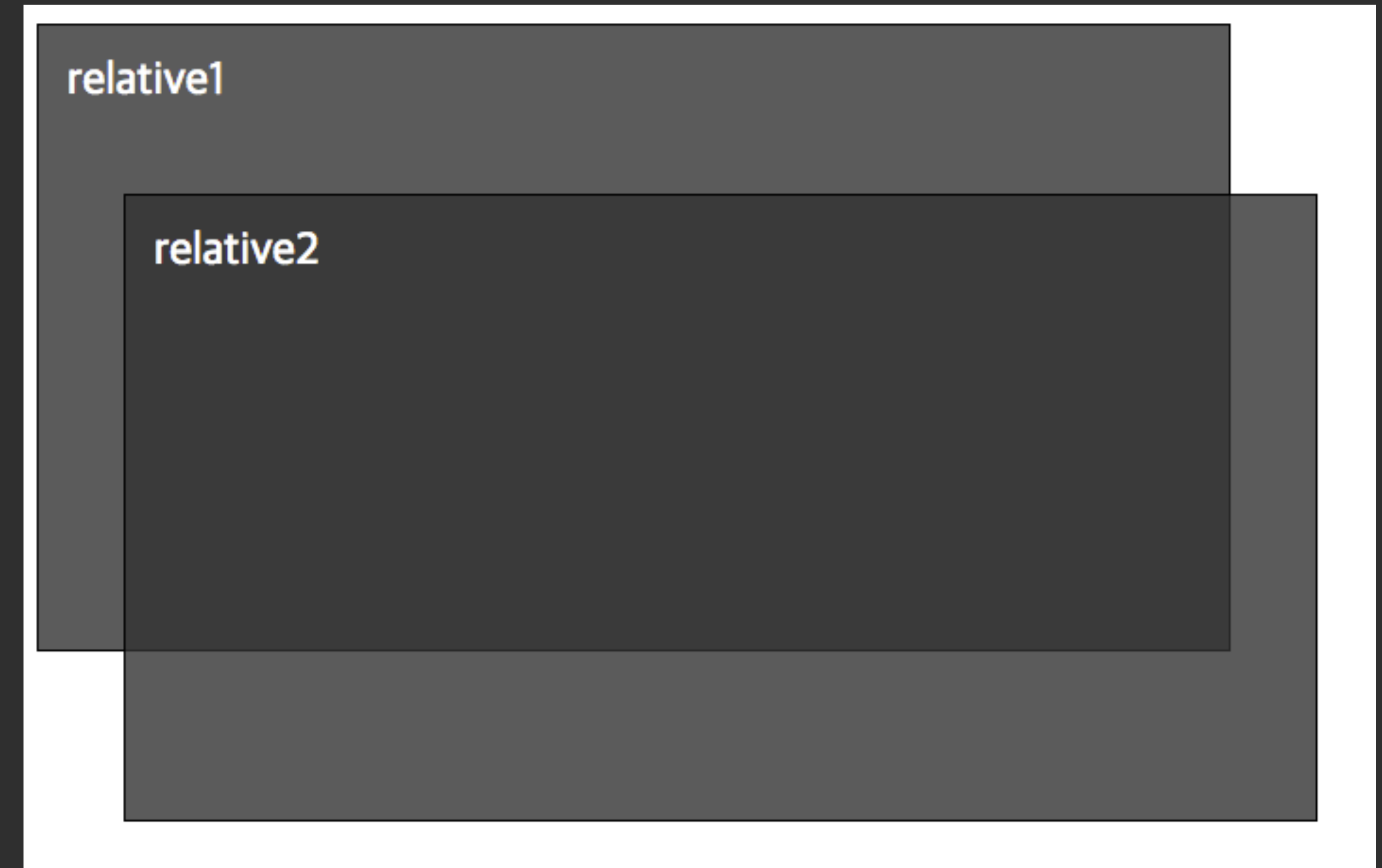
relative

HTML

```
<div class="relative1">  
  relative1  
  <div class="relative2">relative2</div>  
</div>
```

CSS

```
div {  
  width: 400px;  
  height: 200px;  
  padding: 10px;  
  border: 1px solid black;  
  color: white;  
  background-color: rgba(50,50,50,0.8);  
}  
.relative1 {  
  position: relative;  
}  
.relative2 {  
  position: relative;  
  top: 30px;  
  left: 20px;  
}
```



relative 포지션은 자신의 위치를 기준으로 정렬됩니다

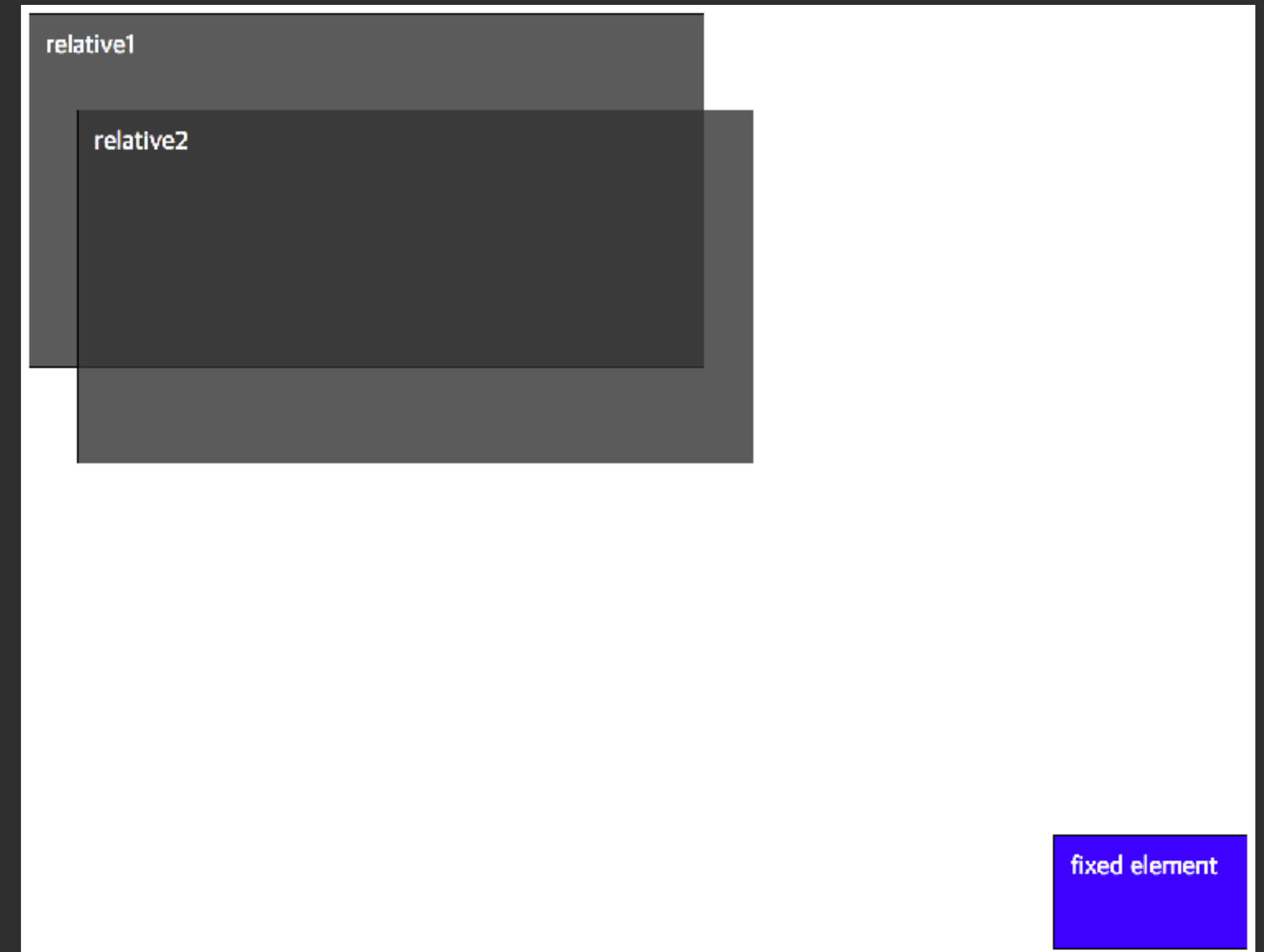
# position

fixed

HTML `<div class="fixed">fixed element</div>`

CSS

```
.fixed {  
  position: fixed;  
  width: 100px;  
  height: 50px;  
  background-color: blue;  
  right: 10px;  
  bottom: 10px;  
}
```



fixed포지션은 뷰포트(표시영역)를 기준으로 정렬됩니다

# position

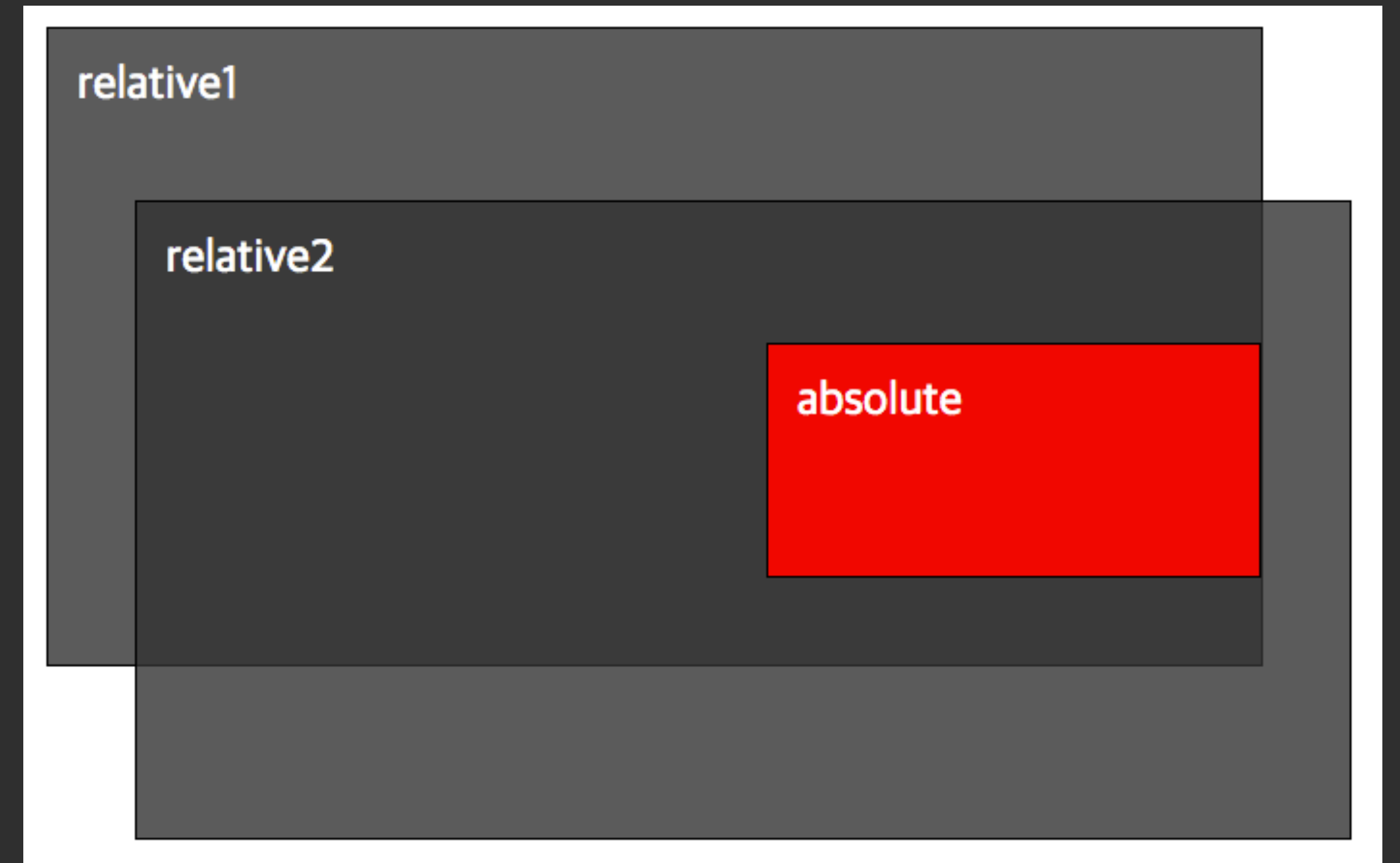
absolute

HTML

```
<div class="relative1">  
  relative1  
  <div class="relative2">relative2</div>  
  <div class="absolute">absolute</div>  
</div>  
<div class="fixed">fixed element</div>
```

CSS

```
.absolute {  
  position: absolute;  
  width: 150px;  
  height: 60px;  
  background-color: red;  
  color: white;  
  right: 0;  
  bottom: 30px;  
}
```



absolute 포지션은 static이 아닌  
가장 가까운 부모를 기준으로 정렬됩니다

# CSS 가운데 정렬

CSS Center positioning

CSS를 사용해서 레이아웃을 가운데 정렬하는 법을 배웁니다

# 가로 가운데 정렬

부모의 가운데로 정렬하는 법



전체 크기가 정해져 있지 않을 경우, 내용의 width만 지정한 후 좌/우 여백은 auto로 같게 처리해줍니다

# 가로/세로 가운데 정렬

부모의 가로/세로 가운데 정렬하는 법

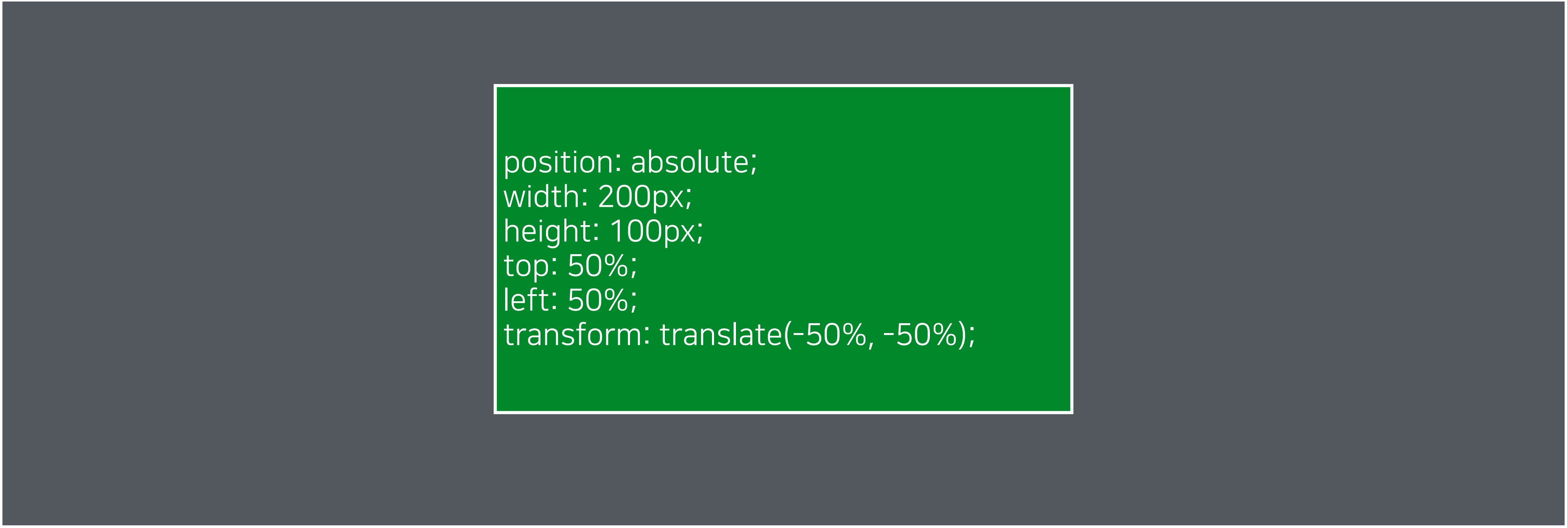
height: 부모요소의 height  
line-height: 부모요소의 height

width: 500px;  
margin: 0 auto;

부모요소의 height와 line-height의 값이 같을 경우, 내부의 요소들은 세로 가운데로 정렬됩니다

# 가로/세로 가운데 정렬

부모의 가로/세로 가운데 정렬하는 법



```
position: absolute;  
width: 200px;  
height: 100px;  
top: 50%;  
left: 50%;  
transform: translate(-50%, -50%);
```

요소를 absolute포지션으로 설정한 뒤, 상단과 왼쪽에서 각각 부모의 50%만큼 이동 후 자신의 높이와 가로의 -50%만큼을 다시 위와 왼쪽으로 이동합니다.

transform을 사용할 경우, 픽셀이 깨져보일 수 있음.

# Semantic Tag

for Document structure

HTML5의 시맨틱 태그



# 시맨틱 태그는?

Semantic tag

웹 사이트에서 자주 사용하는 문서 구조를 태그로 만든 것

검색엔진, 시각장애인용 스크린리더, 개발자에게 문서의 각 부분이 어떤 역할을 하는지 쉽게 알려줌

# 시맨틱 태그의 종류

Semantic tag types

header

머릿말, 페이지 맨 위나 왼쪽에 삽입

nav

내비게이션 링크, 가로/세로 형태의 메뉴에 사용

section

콘텐츠 영역

article

콘텐츠 내용

aside

본문 이외의 내용 (메인 내용에 영향을 주지 않는 링크 등)

footer

꼬릿말, 제작자 및 저작권 정보 표시

# Sass

Syntactically Awesome Stylesheet

# Sass는?

Syntactically Awesome Stylesheet

## CSS전처리기 (Pre-processor)

Sass와 같은 'CSS확장 언어'의 파일을 웹 브라우저에서 해석할 수 있는  
css파일로 만들어주는 처리기

# Sass 기본구조

Sass basic structure

```
div.container {  
  padding: 15px;  
  margin: 0;  
  
  > p#main-title {  
    font-size: 16px;  
    font-weight: bold;  
  }  
  
  > .fixed {  
    position: fixed;  
    bottom: 10px;  
    right: 10px;  
  }  
}
```

# Sass의 출력 스타일 - expanded

Sass output style

Sass

```
div.container {  
  padding: 15px;  
  margin: 0;  
  
  > p#main-title {  
    font-size: 16px;  
    font-weight: bold;  
  }  
  
  > .fixed {  
    position: fixed;  
    bottom: 10px;  
    right: 10px;  
  }  
}
```

CSS

```
div.container {  
  padding: 15px;  
  margin: 0;  
}  
  
div.container p#main-title {  
  font-size: 16px;  
  font-weight: bold;  
}  
  
div.container .fixed {  
  position: fixed;  
  bottom: 10px;  
  right: 10px;  
}
```

# Sass의 출력 스타일 - nested

Sass output style

Sass

```
div.container {  
  padding: 15px;  
  margin: 0;  
  
  > p#main-title {  
    font-size: 16px;  
    font-weight: bold;  
  }  
  
  > .fixed {  
    position: fixed;  
    bottom: 10px;  
    right: 10px;  
  }  
}
```

CSS

```
div.container {  
  padding: 15px;  
  margin: 0; }  
div.container p#main-title {  
  font-size: 16px;  
  font-weight: bold; }  
div.container .fixed {  
  position: fixed;  
  bottom: 10px;  
  right: 10px; }
```

# Sass의 출력 스타일 - compact

Sass output style

Sass

```
div.container {  
  padding: 15px;  
  margin: 0;  
  
  > p#main-title {  
    font-size: 16px;  
    font-weight: bold;  
  }  
  
  > .fixed {  
    position: fixed;  
    bottom: 10px;  
    right: 10px;  
  }  
}
```

CSS

```
div.container { padding: 15px; margin: 0; }  
div.container p#main-title { font-size: 16px; font-weight: bold; }  
div.container .fixed { position: fixed; bottom: 10px; right: 10px; }
```



# Sass의 출력 스타일 - compressed

Sass output style

Sass

```
div.container {  
  padding: 15px;  
  margin: 0;  
  
  > p#main-title {  
    font-size: 16px;  
    font-weight: bold;  
  }  
  
  > .fixed {  
    position: fixed;  
    bottom: 10px;  
    right: 10px;  
  }  
}
```

CSS

```
div.container{padding:15px;margin:0}div.container p#main-title{font-size:16px;font-weight:bold}div.container .fixed{position:fixed;bottom:10px;right:10px}
```

# sass-autocompile

atom package

atom 패키지 설치

sass-autocompile

# sass-autocompile

atom packages - Settings

Compile on Save

Compile with 'compressed'

Compile with 'compact'

Compile with 'nested'

Compile with 'expanded'

저장시 자동 컴파일

Compressed 방식 파일 생성

Compact 방식 파일 생성

Nested 방식 파일 생성

Expanded 방식 파일 생성

# sass-autocompile

atom packages - Settings

상대 경로 지정  
'../css/\$1.css'

# Sass 문법

Sass syntax

Sass 사용법에 대해 간단히 알아봅니다

# 주석

Comments

//주석내용

또는

/\* 주석내용 \*/ (CSS기본)  
을 사용합니다

# 주석

Comments

```
// 한 줄 주석은 Sass에서만 지원하며,  
// CSS컴파일 시 나타나지 않습니다.  
div.container {  
    padding: 15px;  
    margin: 0;  
  
    /* 기존에 CSS에서 사용하는 주석 형태는 컴파일 시에도 나타납니다 */  
    p#main-title {  
        font-size: 16px;  
        font-weight: bold;  
    }  
    /*  
        여러 줄 주석은  
        기존 CSS주석 형태로만 사용 가능합니다  
    */  
    .fixed {  
        position: fixed;  
        bottom: 10px;  
        right: 10px;  
    }  
}
```

# 주석

Comments

```
@charset "UTF-8";
div.container {
    padding: 15px;
    margin: 0;
    /* 기존에 CSS에서 사용하는 주석 형태는 컴파일 시에도 나타나게 됩니다 */
    /*
        여러 줄 주석은
        기존 CSS주석 형태로만 사용 가능합니다
    */
}

div.container p#main-title {
    font-size: 16px;
    font-weight: bold;
}

div.container .fixed {
    position: fixed;
    bottom: 10px;
    right: 10px;
}
```



# 중첩

Nested

CSS에서의 .container > #page를 뜻합니다

CSS에서의 .container > #page p.section-title을 뜻합니다

```
.container {  
  width: 1100px;  
  margin: 0 auto;  
  
  > #page {  
    width: 800px;  
    border: 1px solid #eee;  
    padding: 10px;  
  
    p.section-title {  
      font-size: 20px;  
      text-align: center;  
    }  
  
    p.section-content {  
      font-size: 12px;  
      color: #999;  
    }  
  }  
}
```

# 부모 참조 선택자 (&)

Referencing parent selectors

Sass

&은 자신의  
부모선택자를  
참조합니다

```
#page {  
  width: 800px;  
  border: 1px solid #eee;  
  padding: 10px;  
  
  a {  
    text-decoration: none;  
  
    &:hover {  
      color: red;  
    }  
  
    .link-container & {  
      font-size: 30px;  
    }  
  }  
}
```

& 앞에 선택자가 있을 경우,  
해당 선택자 다음에 현재 위치에서의 부모선택자를 참조합니다

CSS

```
#page {  
  width: 800px;  
  border: 1px solid #eee;  
  padding: 10px;  
}  
  
#page a {  
  text-decoration: none;  
}  
  
#page a:hover {  
  color: red;  
}  
  
.link-container #page a {  
  font-size: 30px;  
}
```

# 중첩 속성

Nested properties

Sass

```
.container {  
  margin: {  
    left: auto;  
    right: auto;  
  }  
}
```

속성 다음에 -(dash)  
로 이어지는 속성은  
아래로 내려  
작성할 수 있습니다

CSS

```
.container {  
  margin-left: auto;  
  margin-right: auto;  
}
```

# 선택자 상속 (@extend)

Selector inheritance

Sass

```
.btn {  
  background-color: #cdcdcd;  
  font-weight: bold;  
  color: white;  
  padding: 5px 20px;  
}  
.btn-ok {  
  @extend .btn;  
  background-color: #d9edf7;  
}  
.btn-cancel {  
  @extend .btn;  
  background-color: #bbb;  
}
```

CSS

```
.btn, .btn-ok, .btn-cancel {  
  background-color: #cdcdcd;  
  font-weight: bold;  
  color: white;  
  padding: 5px 20px;  
}  
.btn-ok {  
  background-color: #d9edf7;  
}  
.btn-cancel {  
  background-color: #bbb;  
}
```

상속받은 선택자가 .btn의 설정을 함께 공유합니다

상속을 사용하면 같은 속성에서 일부만 바뀐 요소를 쉽게 컨트롤 할 수 있습니다.

# 대체 선택자 (%)

Placeholder selector

Sass

```
%btn {
  background-color: #cdcdcd;
  font-weight: bold;
  color: white;
  padding: 5px 20px;
}

.btn-ok {
  @extend %btn;
  background-color: #d9edf7;
}

.btn-cancel {
  @extend %btn;
  background-color: #bbb;
}
```

대체선택자로 선언된 구문은 CSS로 해석되지 않습니다

%btn에 지정한 속성은 CSS에 나타나지 않습니다

CSS

```
.btn-ok, .btn-cancel {
  background-color: #cdcdcd;
  font-weight: bold;
  color: white;
  padding: 5px 20px;
}

.btn-ok {
  background-color: #d9edf7;
}

.btn-cancel {
  background-color: #bbb;
}
```

# 변수 (\$)

## Variables

```
$padding: 10px;  
$bg-color: #ececec;  
$title-font-weight: bold;
```

변수명은 \$로 시작합니다

```
div.container {  
  background-color: $bg-color;  
  padding: $padding;  
  margin: 0;
```

자주 쓰이는 color값을 저장합니다

```
p#main-title {  
  font-size: 16px;  
  font-weight: $title-font-weight;  
}
```

중복으로 쓰이는 값에 유용합니다

```
p#sub-title {  
  font-size: 14px;  
  font-weight: $title-font-weight;  
}
```

```
.fixed {  
  position: fixed;  
  padding: $padding;  
  bottom: 10px;  
  right: 10px;  
}
```

# Sass파일 호출

## Import

현재 파일 위치를 기준으로 불러옵니다

```
@import 'variables';

%btn {
  background-color: $brand-primary;
  font-weight: bold;
  color: white;
  padding: 5px 20px;
}
```

CSS파일을 불러올 때는 확장자를 입력하여야 하지만, Sass파일을 불러올 때는 확장자를 입력하지 않아도 됩니다.

\*\*\_로 시작하는 파일은 CSS파일로 컴파일 되지 않습니다. import시에만 사용하는 파일은 \_로 네이밍하세요