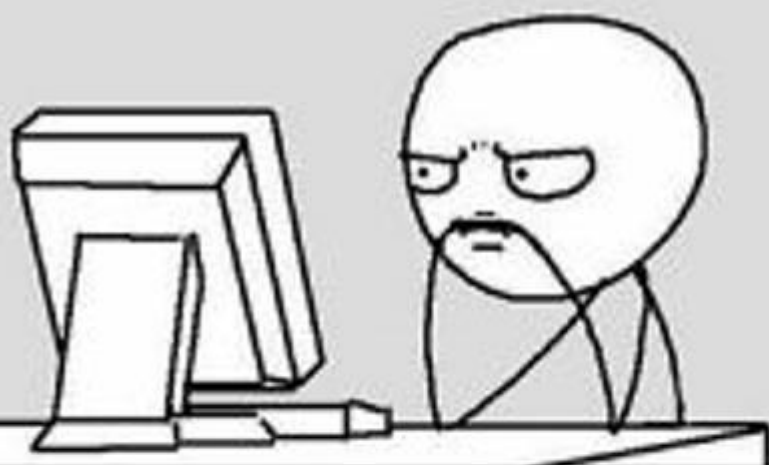
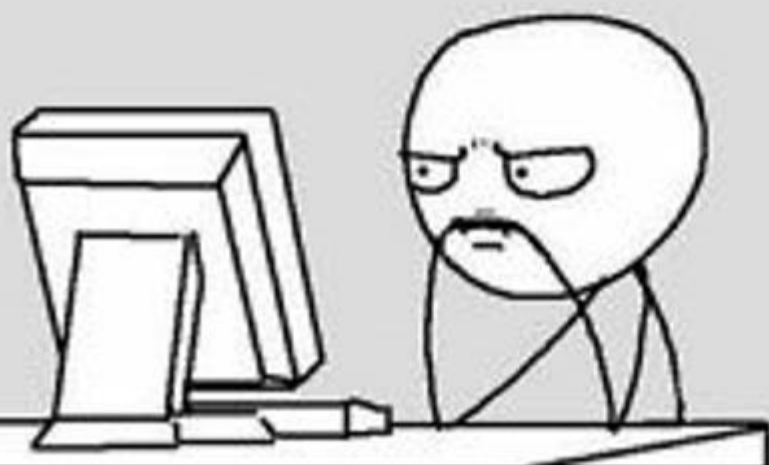


It doesn't work..... why?



It works..... why?



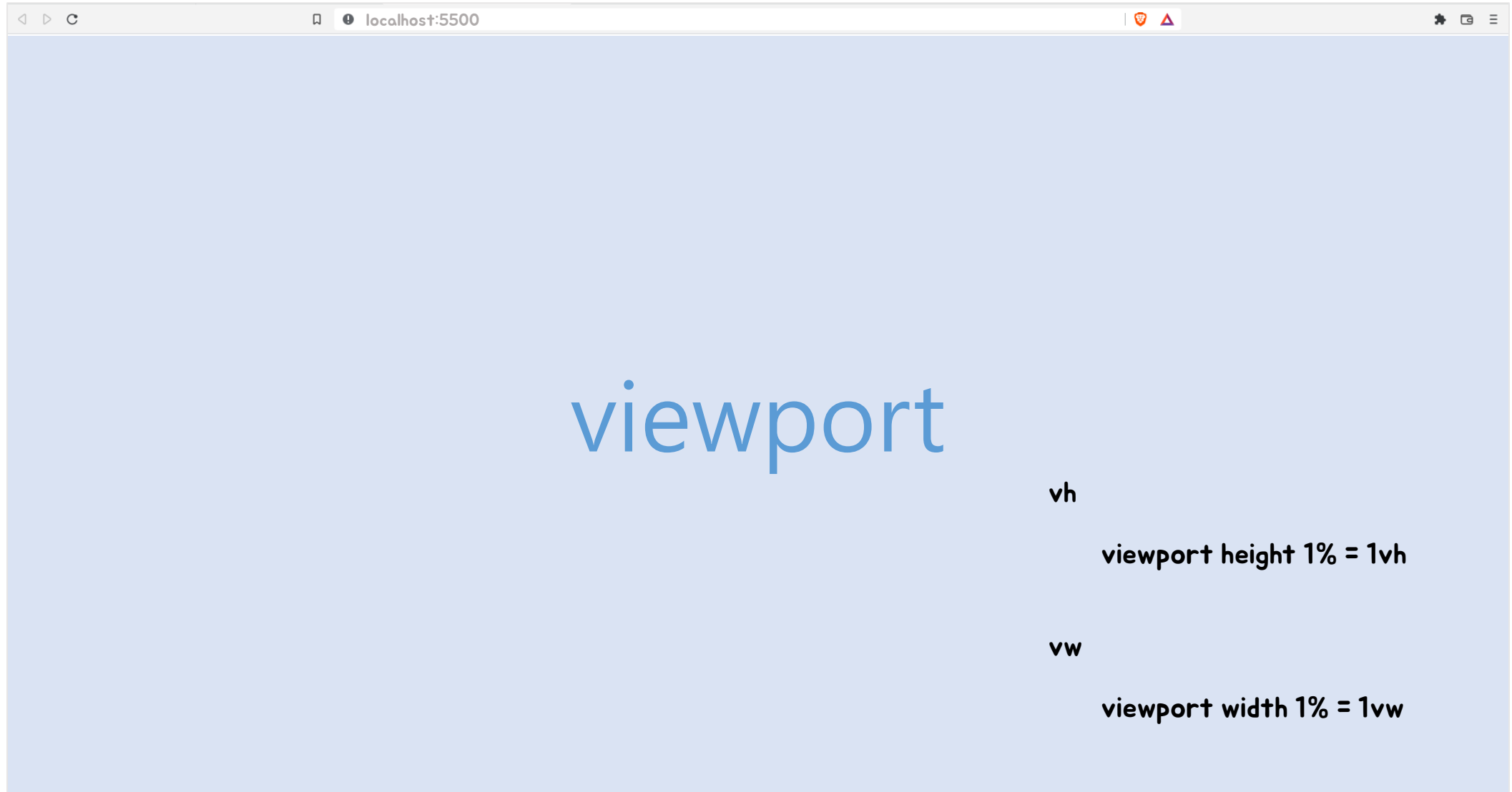
CSS

Cascading Style Sheet



CSS

Viewport



```
선택자 {  
  
    속성: 값;  
  
}
```

Type HTML 태그 직접 선택

Class Class명으로 선택 **.클래스명 { 속성: 값; }**
 Class는 여러 개를 가질 수 있습니다

ID ID명으로 선택 **#클래스명 { 속성: 값; }**
 id는 1개만 가질 수 있습니다

.box-1.box-2

.box-1 .box-2

위의 2개의 선택은 다릅니다.

자식/자손/형제 선택자와 연관 있습니다.

CSS

선택자 우선순위

순위		금	은	동	합계
1	 미국	39	41	33	113
2	 중국	38	32	18	88
3	 일본	27	14	17	58
4	 영국	22	21	22	65
5	 ROC	20	28	23	71
6	 호주	17	7	22	46
7	 네덜란드	10	12	14	36
8	 프랑스	10	12	11	33
9	 독일	10	11	16	37

올림픽 순위는 합계 점수가 중요한 것이 아니라
금 >>> 은 >> 동 에 따라 순위 배정

CSS

선택자 우선순위

동일한 태그가 있다면 나중에 쓴 부분이 적용됩니다.



ID 선택자



Class 선택자
가상 선택자



Tag 선택자

올림픽 점수 내는 것과 동일하다고 생각하면 쉽습니다.

Inline Style

우선순위가 강하게 적용되지만, 지양하는 방법

!important

우선순위가 강하게 적용되지만, 지양하는 방법

CSS Selector

자식 선택자
Child

parent>child

자손 선택자
Descendant

parent descendants

형제 선택자
Sibling

parent+_sibling

parent~_sibling

구조적 가상 클래스 선택자

Structural Pseudo-classes

first-child

last-child

nth-child(n)

n에 점화식 사용 가능

동적 가상 클래스 선택자

User Action Pseudo-classes

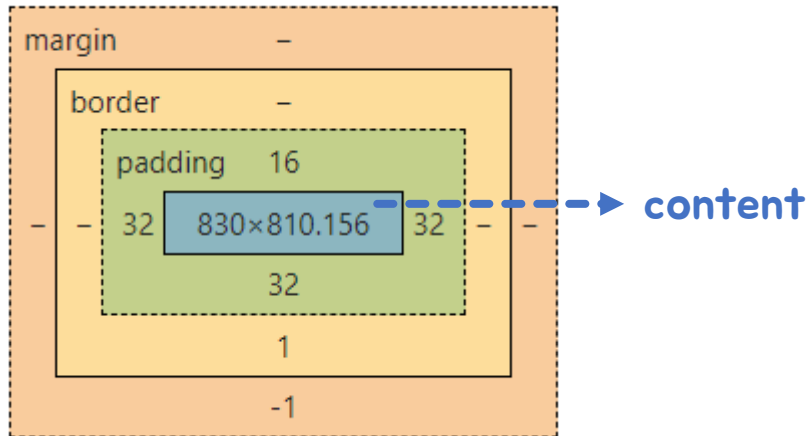
hover

focus

active

CSS

Box model



content

가로 width
세로 height

padding

안쪽 여백
content와 border 사이의 공백을 나타냄

border

테두리를 나타냄

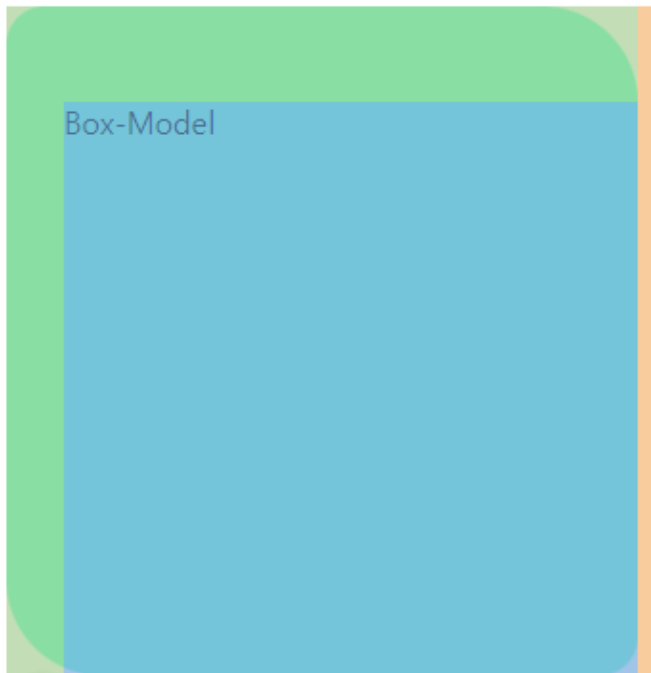
```
div {  
  border: 1px solid blue;  
}
```

굵기 스타일 색상

`border: none;`
선을 안 나타내고 싶을 때 사용

margin

바깥 여백
요소와 요소 사이의 간격을 나타냄



div	330 × 350
Color	■ #000000
Font	16px "Malgun Gothic"
Background	■ #7FFFD4
Padding	50px 0px 0px 30px
ACCESSIBILITY	
Contrast	Aa 17.15 ✓
Name	
Role	generic
Keyboard-focusable	⊘

선택자 {

box-sizing: content-

content-box | border-box

box;

}

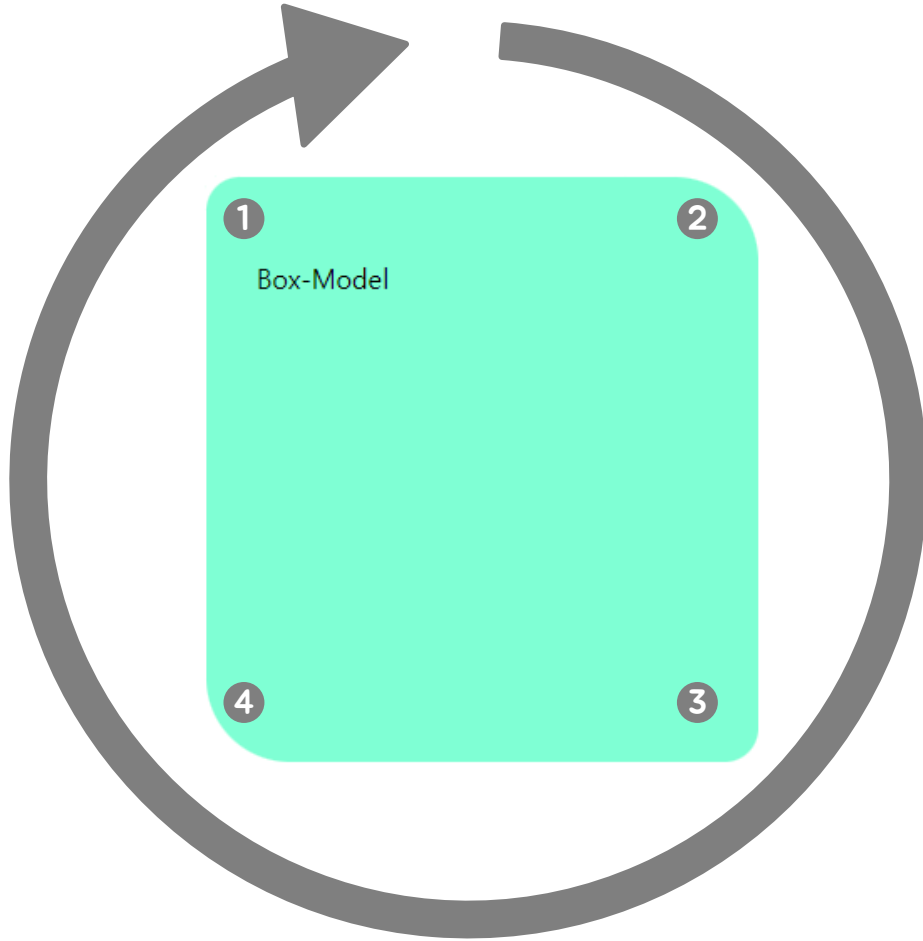
기본으로 content-box로 설정되어 있습니다.

Padding을 주면 해당 크기까지 더해서 요소의 크기가 결정됩니다.

크기의 계산을 편리하게 하기 위해서 border-box로 변경하여 사용하는 것을 권장합니다.

CSS

Box model



HTML CODE

```
<div>Box-Model</div>
```

CSS CODE

```
div {  
  width: 300px;  
  height: 300px;  
  background-color: aquamarine;  
  border-radius: 20px 50px;  
  padding: 50px 0px 0px 30px;  
}
```

radius의 경우 모서리 1&3 2&4 순

속기형, Shorthand

2개일 때 TOP & BOTTOM RIGHT & LEFT

4개일 때 TOP RIGHT BOTTOM LEFT



block

inline

inline-block

flex

CSS

display: block

block

미국식[bla:k]  영국식[blɒk] 

명사

1 (단단한) 사각형 덩어리 (→breeze **block**, building **block**, cinder **block**)

a **block** of ice / concrete / stone 

네모난 얼음 덩어리/콘크리트 덩어리/돌덩어리


2 (특정 목적용) 건물, ...관

a tower **block** 

고층 건물


동사

1 (지나가지 못하게) 막다, 차단하다

After today's heavy snow, many roads are still blocked. 

오늘 내린 폭설로 많은 도로들이 아직 통행이 안 된다.

2 ~의 길, 출구, 시야 등을 막다

One of the guards moved to **block** her path. 

경비원 중 한 명이 나와 그녀의 길을 막았다.

영어사전 다른 뜻 1

따로 width를 선언하지 않으면

부모 width의 context-box 100% 크기

width를 선언시

남은 공간은 margin으로 채움

따로 height 선언하지 않으면

자식 요소의 height의 합 = 부모의 height

CSS

display: inline

inline 웹수집

인라인의, 그때마다 즉시 처리하는, (내연기관이)직렬의, (부품 장치가)일렬로 늘어선

영어사전 다른 뜻 1

사용불가

width, weight, padding-top, padding-bottom,

margin-top, margin-bottom , border-top, border-bottom

CSS

display: inline-block

Block

Span Inline Block

HTML CODE

```
<div class="block-div">Block</div>
<div class="inline-span">Span</div>
<div class="inline-block">Inline Block</div>
```

CSS CODE

```
.inline-block {
  display: inline-block;
  width: 500px;
  background-color: blue;
  color: white;
}

.inline-span {
  display: inline;
  width: 500px;
  background-color: pink;
}

.block-div {
  display: block;
  background-color: green;
}
```

Inline-block은

width와 height를 지정해줄 수 있다.
(지정해주지 않을 경우 inline과 동일)

CSS

typography: font-size

letter-spacing
↔
line-height ↕ 웹 프로그래밍 CSS 배우기 ↖ baseline
font-size

- px — 절대 단위 Absolute unit

- em — 상대 단위 Relative unit

- rem

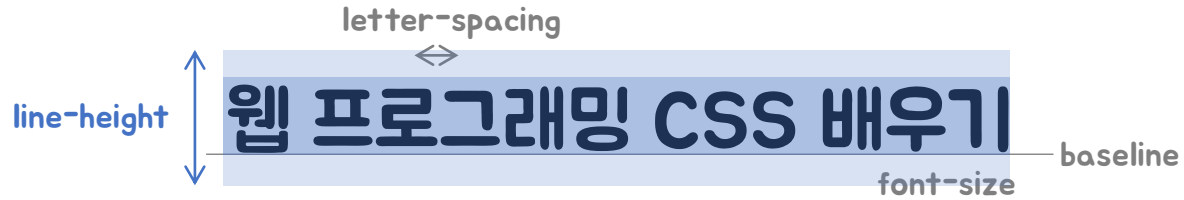
equal to capital M : 실제로 적용된 폰트 사이즈

root em = html em

HTML에게 적용된 font-size 기준 비율

CSS

typography: line-height



- **px**
- **em** line-height 사용시 보편적으로 사용
em으로 사용시에 글이 뒤에 em을 붙이지 않고들 사용합니다.
- **rem**

line-height의 크기가 얼마가 되었든 간에 텍스트는 가운데에 위치하게 됩니다.

CSS

typography: font-weight

letter-spacing
↔

line-height ↑
↓

웹 프로그래밍 CSS 배우기

font-family font-weight font-size baseline

```
font-weight: 500;
```

100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

100단위 regular bold

CSS

typography: color



- colors  lightgreen;
- rgb rgb(126, 224, 129)
- hex #90ee90
- hsl hsl(122, 61%, 69%)

CSS

typography: text-align

left

웹 프로그래밍 CSS 배우기

center

웹 프로그래밍 CSS 배우기

right

웹 프로그래밍 CSS 배우기

CSS

typography: font-family



```
font-family: "Poppins", sans-serif;
```

앞에게 없으면 뒤에꺼 적용 식으로...

webfont

1. Google font와 같은 사이트 방식 이용
2. 직접 폰트 제공

<https://fonts.google.com/>

CSS

typography

text-indent

문장 처음을 들여쓰기

```
text-indent: 50px;
```

text-transform

알파벳과 같이 소문자 | 대문자로 나눠 있는 문자들에게 사용됨

none | **capitalize** | **uppercase** | **lowercase**

앞 글자 단어만 대문자 대문자 처리

소문자 처리

CSS

typography

text-decoration

알파벳과 같이 소문자 | 대문자로 나뉘 있는 문자들에게 사용됨

none | underline | line-through | overline

웹 프로그래밍 CSS배우기

~~웹 프로그래밍 CSS배우기~~

웹 프로그래밍 CSS배우기

font-style

normal | italic | oblique

텍스트 기울임

CSS

background

background-color

hex | rgb | rgba

background-image

url(경로)

1. 파일 경로
2. URL

background-size

contain | cover | custom
자체적 명시

background-repeat

repeat | no-repeat

background-position


위치 설정

float

미국식[flout]  영국식[flaut] 


동사

1 (물 위나 공중에서) 떠[흘러]가다[떠돌다] (=drift)

A group of swans floated by. 

한 무리의 백조들이 물 위를 떠갔다.

2 (가라앉지 않고 물에) 뜨다

Wood floats. 

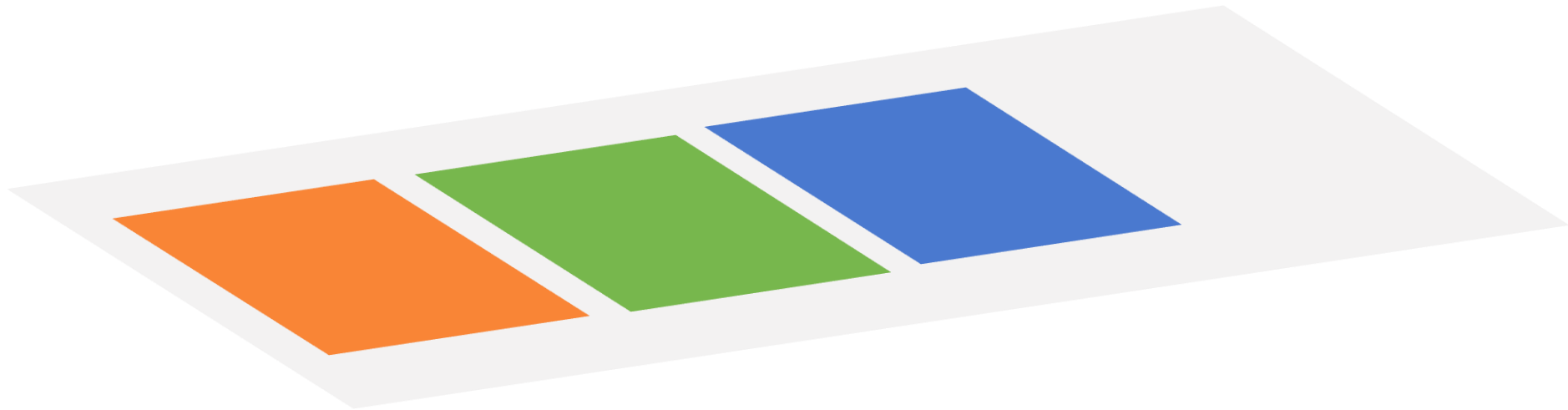
나무는 물에 뜬다.

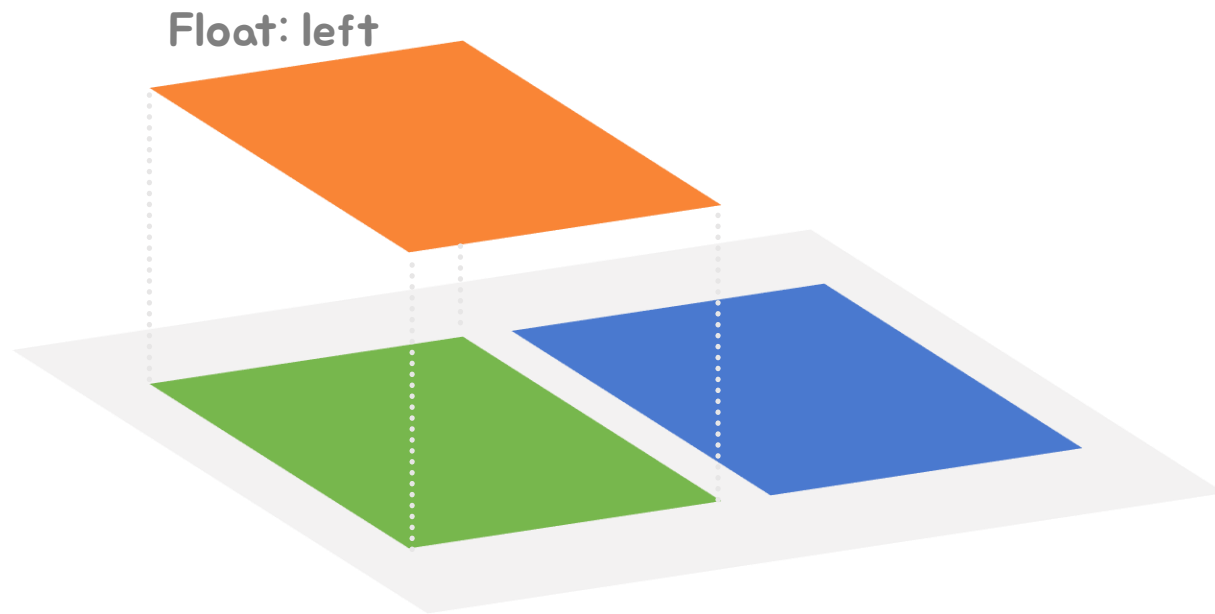
float를 사용하면 Inline / Inline Block / Block → Block으로 변경됨

Block의 속성은 가지게 되나,

Block의 기본 width를 부모 width와 같게 가지는 속성은 못 가지게 됩니다.

자동으로 생기는 여유공간을 채우던 margin 속성도 사라집니다.

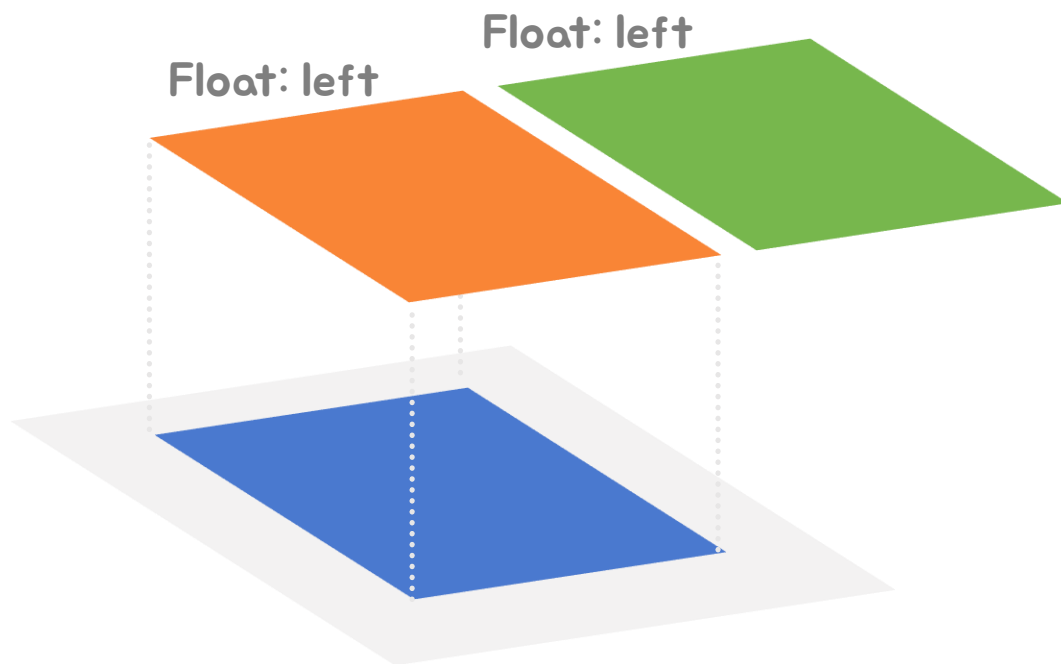




parent 부분에 height를 지정하여 고정 크기를 가지게 하면 문제가 생깁니다.

고정 크기를 가지지 않으면 float된 영역은 나갔다고 생각하기에 부모 요소는 해당 크기를 줄이게 됩니다.

그 다음 요소를 float로 띄우게 되면 옆에 들어갈 수 있는 공간이 있는지 확인 후에
들어갈 수 있는 공간이 없다면 아래에 놓이게 됩니다.
그러면서 해당 요소도 나갔다고 부모 요소는 판단하여 해당 영역을 포함하지 않습니다.



CODE와 함께 변화를 살펴봐 주세요.

다음 페이지부터 내용 설명

HTML CODE

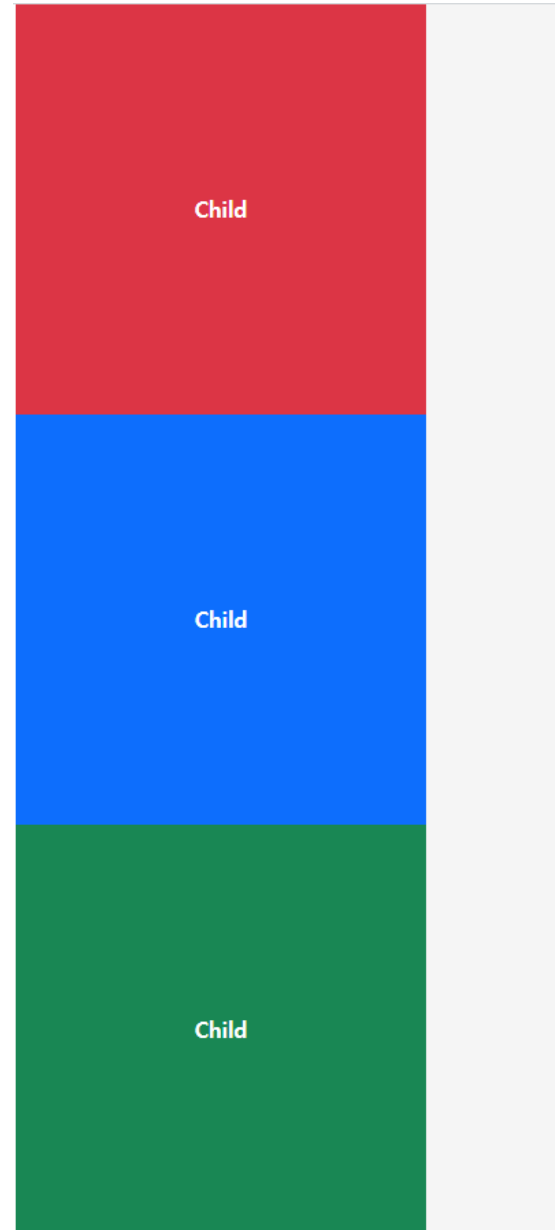
```
<div class="parent">
  <div class="child red">Child</div>
  <div class="child blue">Child</div>
  <div class="child green">Child</div>
</div>
```

CSS CODE

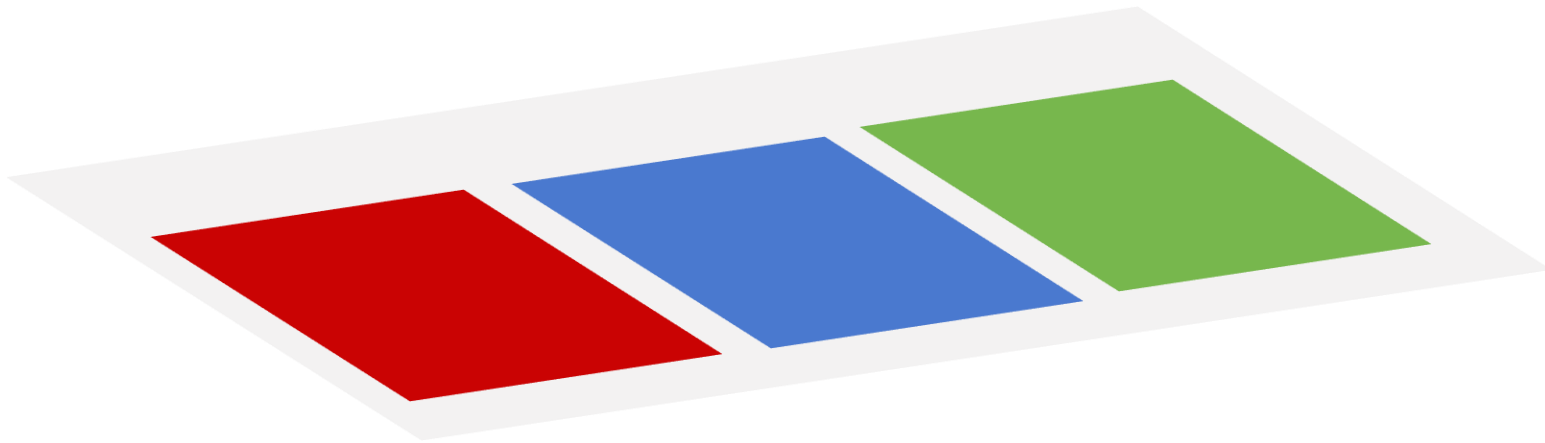
```
1  * {
2    box-sizing: border-box;
3    margin: 0;
4  }
5
6  .parent {
7    width: 400px;
8    margin: 0 auto;
9    background-color: whitesmoke;
10 }
11
12 .child {
13   width: 300px;
14   height: 300px;
15   line-height: 300px;
16   color: white;
17   font-weight: bold;
18   text-align: center;
19 }
20
21 .red {
22   background-color: #dc3545;
23 }
24
25 .blue {
26   background-color: #0d6efd;
27 }
28
29 .green {
30   background-color: #198754;
31 }
```

브라우저의 초기화면

설명을 돕기 위해서 code에서
parent의 width는 400px으로 두었습니다.



초기 브라우저의 모습을 그림으로 보기 좋게 나타내면 다음과 같습니다.



단계별 확인을 위해 개발자 도구에서 변경하면서 확인하도록 하겠습니다.

첫 번째 요소인 red 부분에 float: left를 선언했을 때입니다.

div.parent 400 x 600

Color ■ #000000

Font 16px "Malgun Gothic"

Background □ #F5F5F5


Margin 0px 698px

ACCESSIBILITY

Name

Role generic

Keyboard-focusable ☹

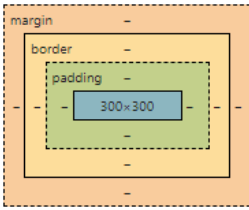


```

<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div class="parent">
      ...
      <div class="child red" style="
        float: left;
        ">Child</div> == $0
      <div class="child blue">Child</div>
      <div class="child green">Child</div>
    </div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>

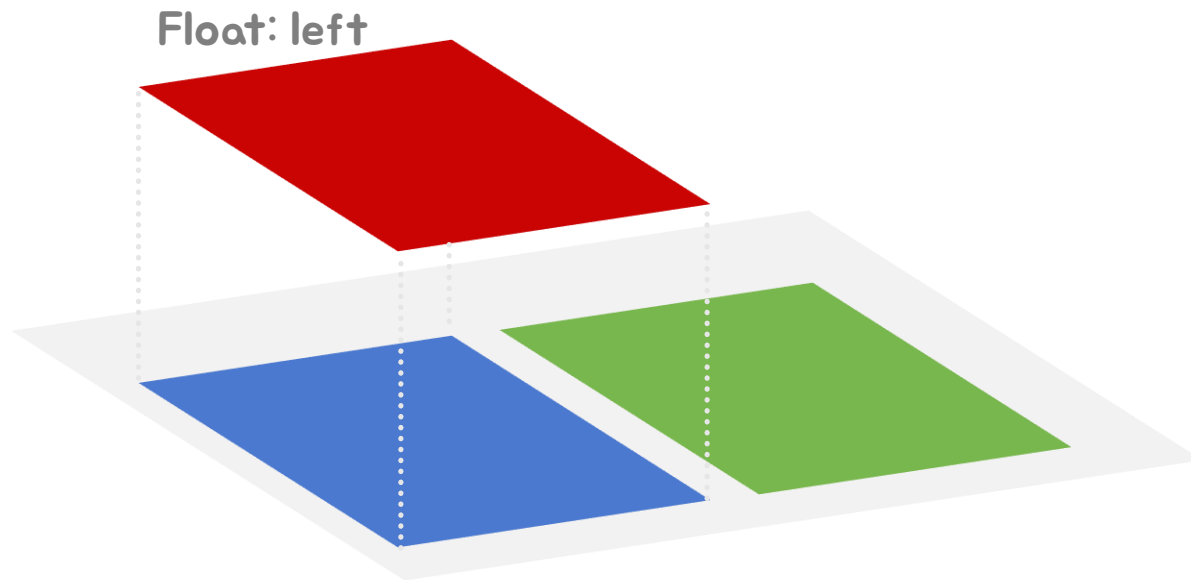
```

Styles	Computed	Layout	Event Listeners	>>
Filter :hov .cls + [icon] [icon]				
element.style { float: left; }				
.red { background-color: #dc3545; }	style.css:21			
.child { width: 300px; height: 300px; line-height: 300px; color: white; font-weight: bold; text-align: center; }	style.css:12			
* { box-sizing: border-box; margin: 0; }	style.css:1			
div { display: block; }	user agent stylesheet			

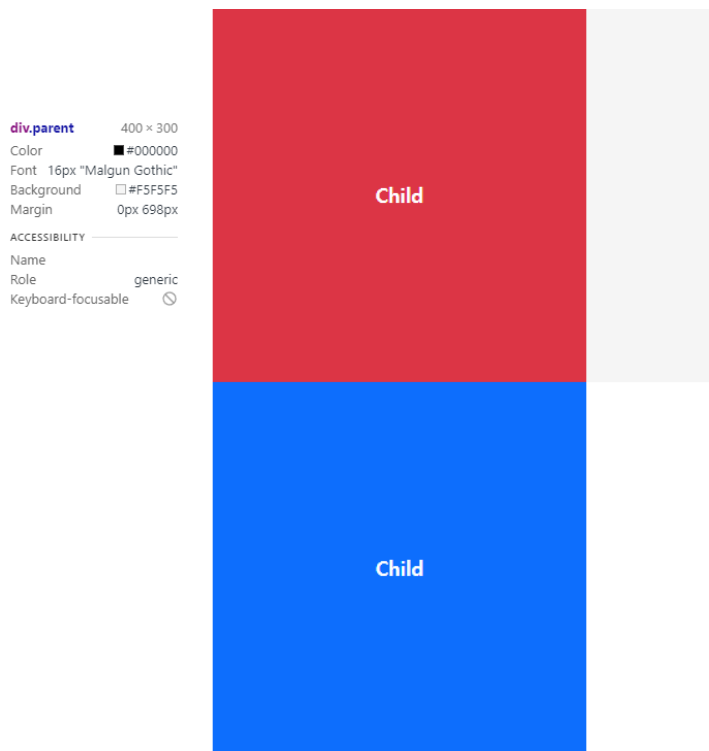


부모가 하나의 child 영역 만큼을 없앤 것을 확인할 수 있습니다.

아까 본 화면을 그림으로 보기 좋게 나타내면 다음과 같습니다.



두번째 요소인 blue 부분에 float: left를 선언했을 때입니다.



```

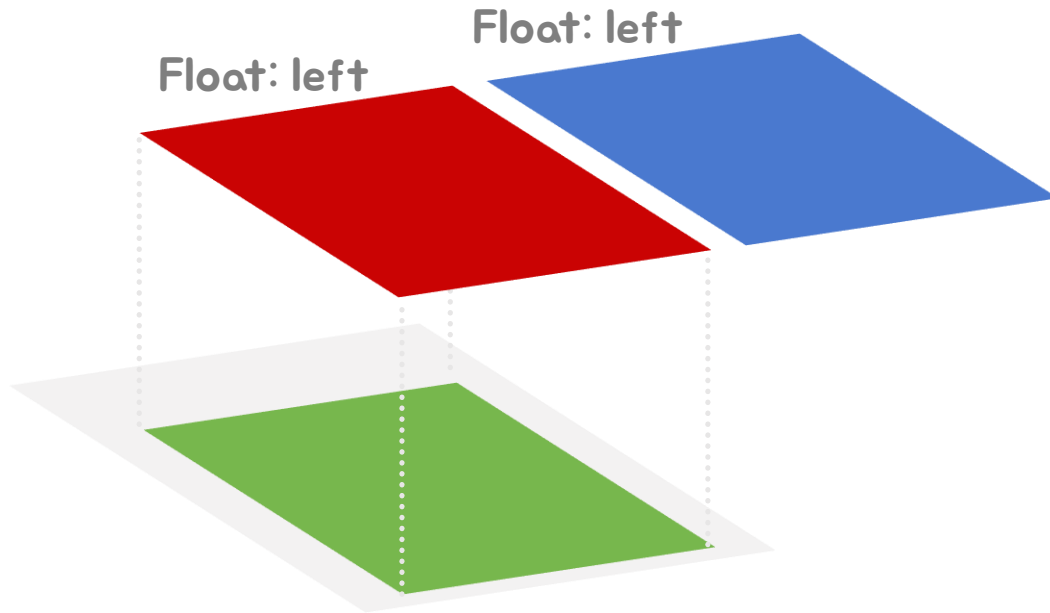
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div class="parent">
      <div class="child red" style="
        float: left;
      ">Child</div>
      <div class="child blue" style="
        float: left;
      ">Child</div>
      <div class="child green">Child</div>
    </div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>

```

Styles	Computed	Layout	Event Listeners	>>
Filter	:hov	.cls	+	[-]
element.style { float: left; }				
.blue { background-color: #0d6efd; }				
.child { width: 300px; height: 300px; line-height: 300px; color: white; font-weight: bold; text-align: center; }				
* { box-sizing: border-box; margin: 0; }				
div { display: block; }				

부모가 또 하나의 child 영역 만큼을 없앤 것을 확인할 수 있습니다.

두번째 요소인 blue 부분에 float: left를 선언했을 때를 그림으로 나타내면 다음과 같습니다.

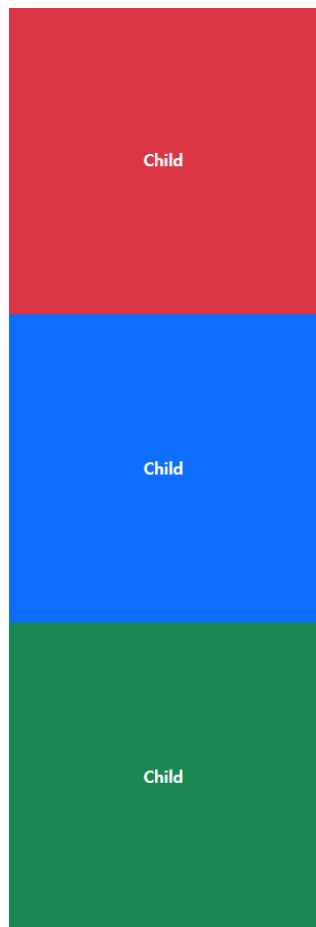


CSS float

마지막 자식 요소인 green부분에 float: left를 선언했을 때입니다.

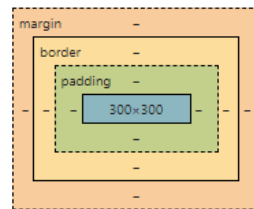
left | right

div.parent 400 × 0



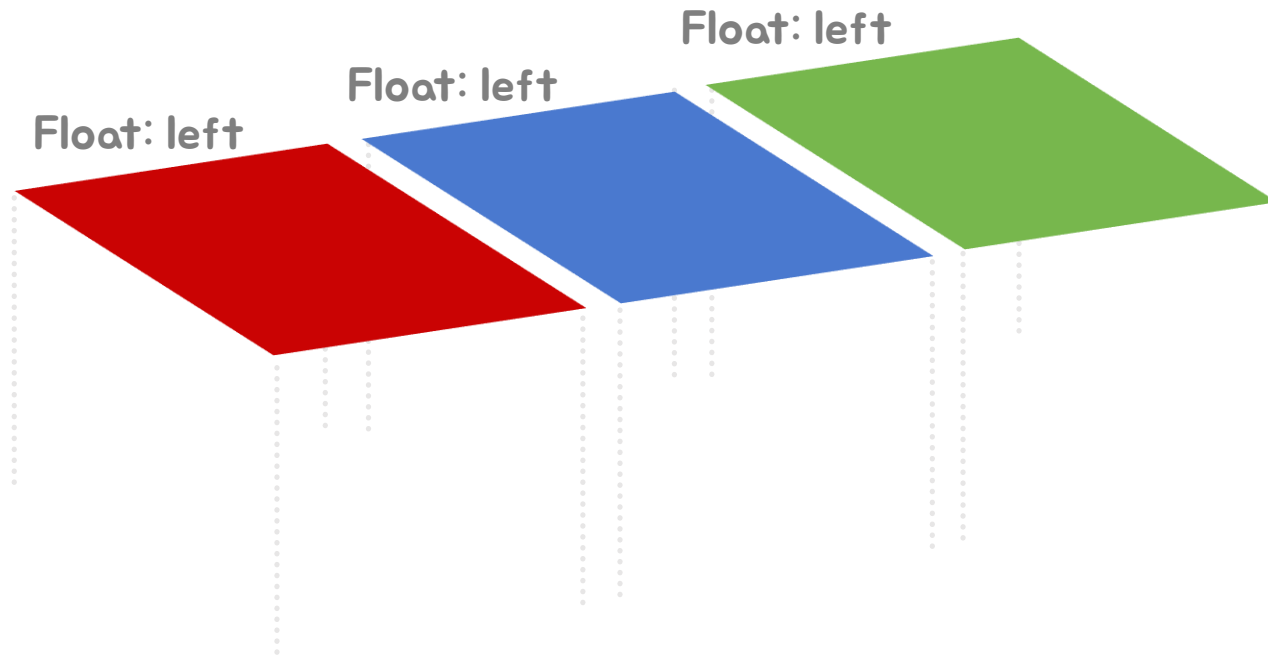
```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div class="parent">
      <div class="child red" style="
        float: left;
      ">Child</div>
      <div class="child blue" style="
        float: left;
      ">Child</div>
      <div class="child green" style="
        float: left;
      ">Child</div> == $0
    </div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

Styles	Computed	Layout	Event Listeners	>>
Filter	:hov	.cls	+	⌵
element.style { float: left; }				
.green { background-color: #198754; }				
.child { width: 300px; height: 300px; line-height: 300px; color: white; font-weight: bold; text-align: center; }				
* { box-sizing: border-box; margin: 0; }				
div { display: block; }				



부모의 height가 0이 된 것을 확인할 수 있습니다.

마지막 자식 요소인 green 부분에 `float: left`를 선언했을 때를 그림으로 나타내면 다음과 같습니다.

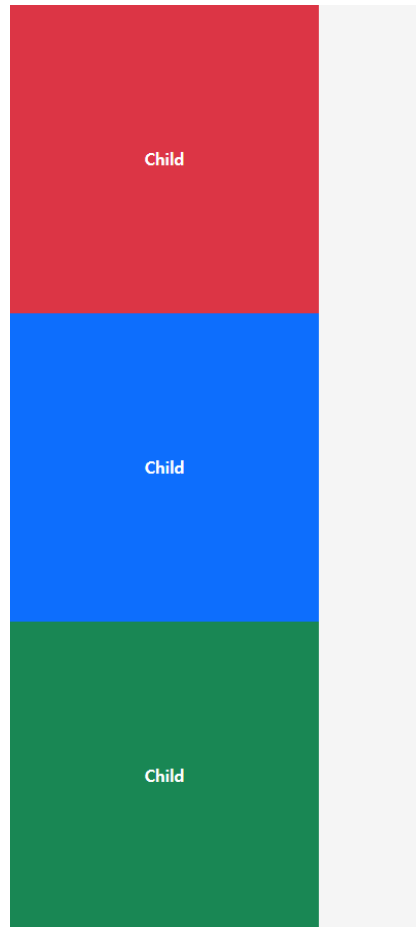


CSS

float

overflow: hidden

부모 요소에 overflow: hidden; 이라고 작성하게 되면 알아서 해당 영역을 잡아주게 됩니다.



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div class="parent" style="
      overflow: hidden;
    ">
      <div class="child red" style="
        float: left;
      ">Child</div>
      <div class="child blue" style="
        float: left;
      ">Child</div>
      <div class="child green" style="
        float: left;
      ">Child</div>
    </div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

Styles Computed Layout Event Listeners >>

Filter :hov .cls + [[]]

element.style {
 overflow: hidden;
}

.parent { style.css:6
 width: 400px;
 margin: 0 auto;
 background-color: whitesmoke;
}

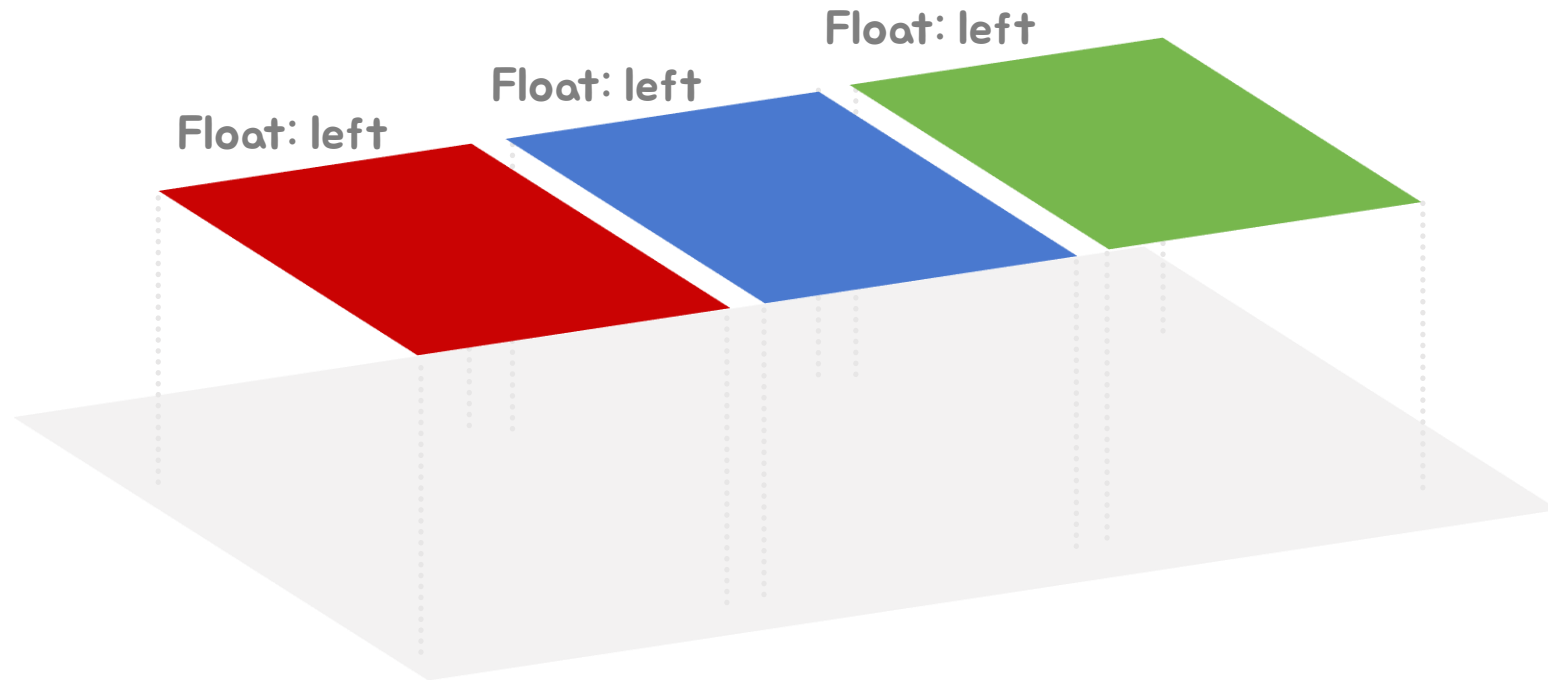
* { style.css:1
 box-sizing: border-box;
 margin: 0;
}

div { user agent stylesheet
 display: block;
}

CSS

float

overflow: hidden



left | right | both

block 요소에만 사용 가능

Clear는 불필요한 div를 만들어서 사용하기도 하지만 이는 너무 불필요하게 사용됩니다.

그래서 이를 해결하기 위해서 가상 요소(pseudo-element)를 사용합니다.

::before / ::after

content 속성이 꼭 필요

```
.parent::after {  
  content: "";  
  display: block;  
  clear: both;  
}
```

CSS 코드에서 해당 부분을 추가해서 진행해주세요.

Child 부분에 float를 주면서 확인을해주세요.

clear는 display가 block요소에만 사용가능하여 display: block 선언해줍니다.

clear 속성값에 따라 기준이 달라지며 float: left로 된 것을 찾으려면 clear: left

Right면 동일하게 right 써주시면 됩니다. both는 left와 right 둘 다 입니다.

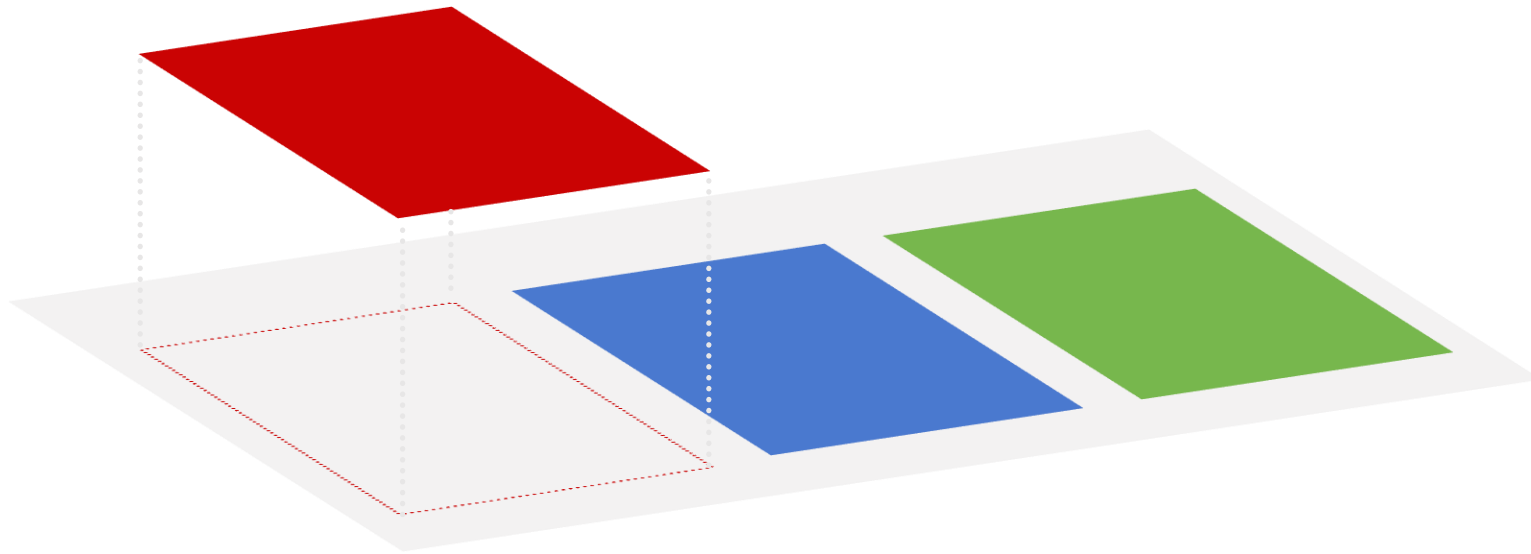
때에 따라 사용이 다르겠지만, 보통 both를 쓰면 편하게 될 거 같습니다.

CSS position

static | relative | absolute | fixed | sticky

static 기본

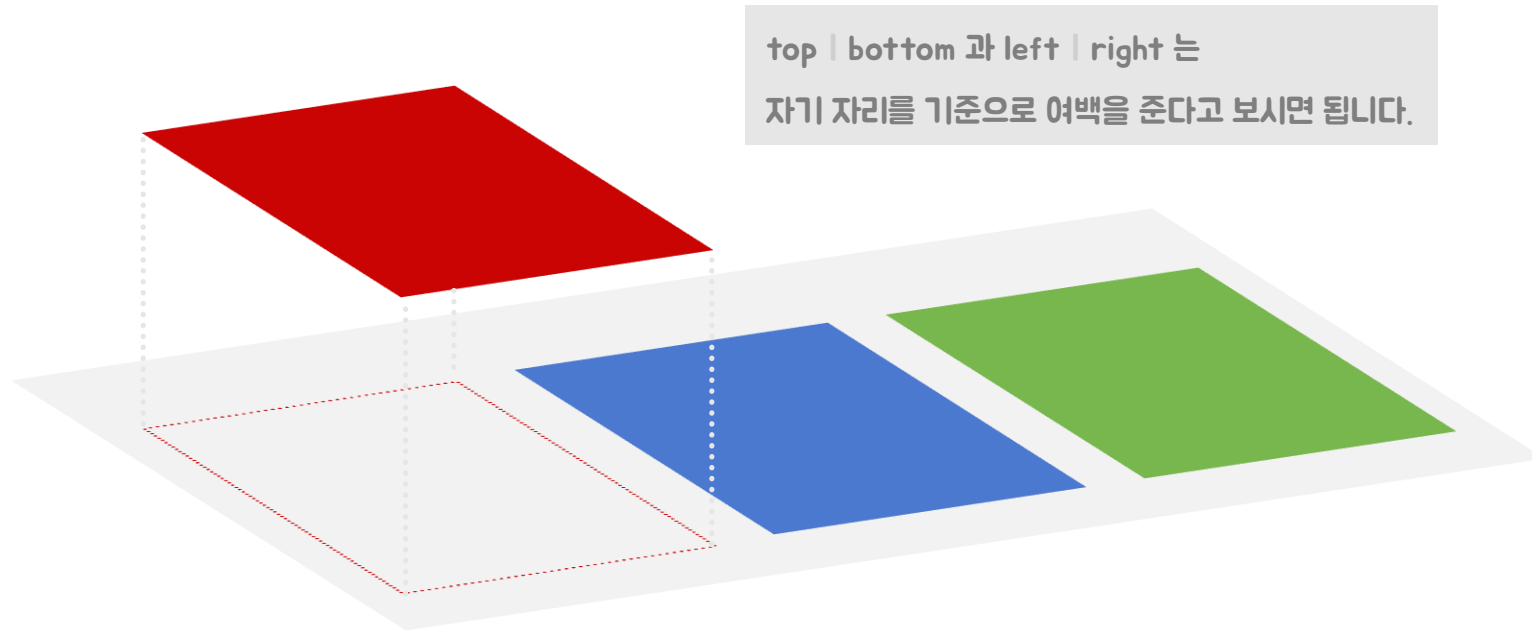
relative Float와 동일하나 위치에 대한 기억은 합니다.



해당 값부터는 `top` | `bottom` | `right` | `left` 와 `z-index` 를 사용할 수 있습니다.

제공된 코드에서 red에서만 `position: relative`를 주게 되면 `float`와 같이 뜨게 되지만,
parent 영역의 공간이 줄어들진 않습니다.

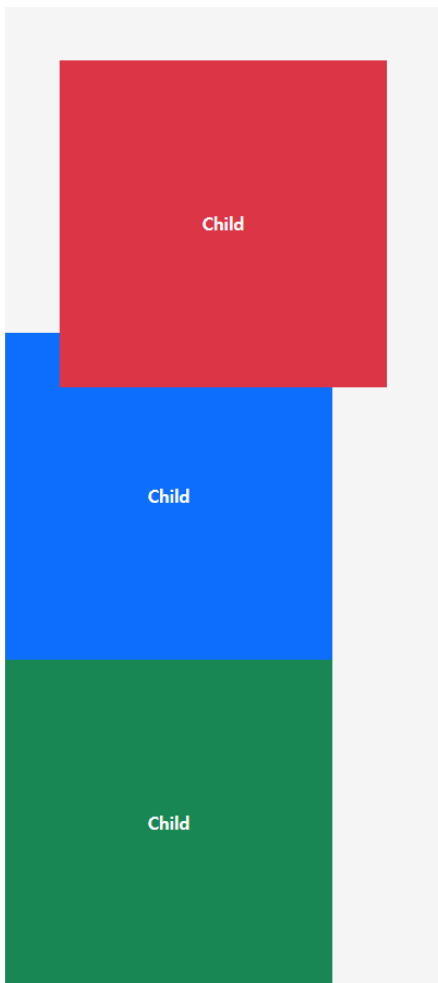
이를 보아 `position`은 자기 자리를 기억한다는 것을 알 수 있습니다.



CSS

position

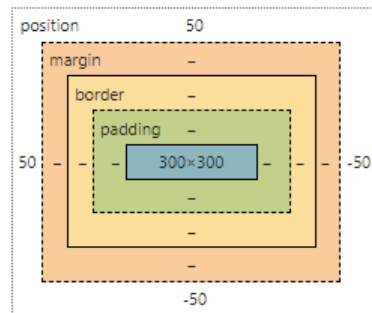
relative



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div class="parent">
      <div class="child red" style="
        position: relative;
        left: 50px;
        top: 50px;
      ">Child</div> == $0
      <div class="child blue">Child</div>
      <div class="child green">Child</div>
    </div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

html body div.parent div.child.red

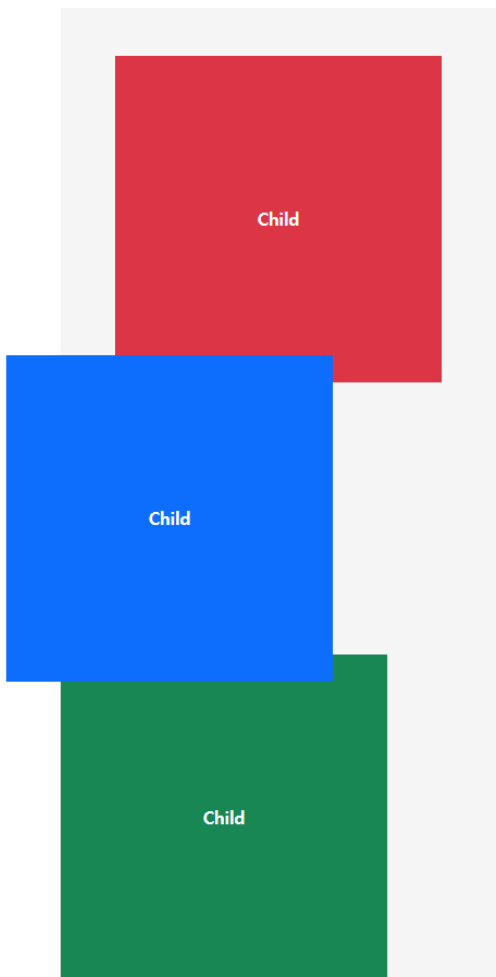
스타일	계산됨	레이아웃	이벤트 리스너	DOM 중단점	»
필터		:hov .cls	+		
element.style {					
position: relative;					
left: 50px;					
top: 50px;					
}					
.red {					
background-color:  #dc3545;					
}					
.child {					
width: 300px;					
height: 300px;					
line-height: 300px;					
color:  white;					
font-weight: bold;					
text-align: center;					
}					
* {					
box-sizing: border-box;					
margin: ▶ 0;					
}					
div {					
display: block;					
}					



CSS

position

relative



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div class="parent">
      <div class="child red" style="
        position: relative;
        left: 50px;
        top: 50px;
      ">Child</div>
      <div class="child blue" style="
        position: relative;
        right: 50px;
        top: 25px;
      ">Child</div> == $0
      <div class="child green">Child</div>
    </div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

html body div.parent div.child.blue

스타일 계산됨 레이아웃 이벤트 리스너 DOM 중단점 >>

필터 :hov .cls +

element.style {
position: relative;
right: 50px;
top: 25px;
}

.blue {
background-color: #0d6efd;
} style.css:25

.child {
width: 300px;
height: 300px;
line-height: 300px;
color: white;
font-weight: bold;
text-align: center;
} style.css:12

* {
box-sizing: border-box;
margin: 0;
} style.css:1

div {
display: block;
} 사용자 에이전트 스타일시트

position 25

margin -

border -

padding -

300x300

-50

50

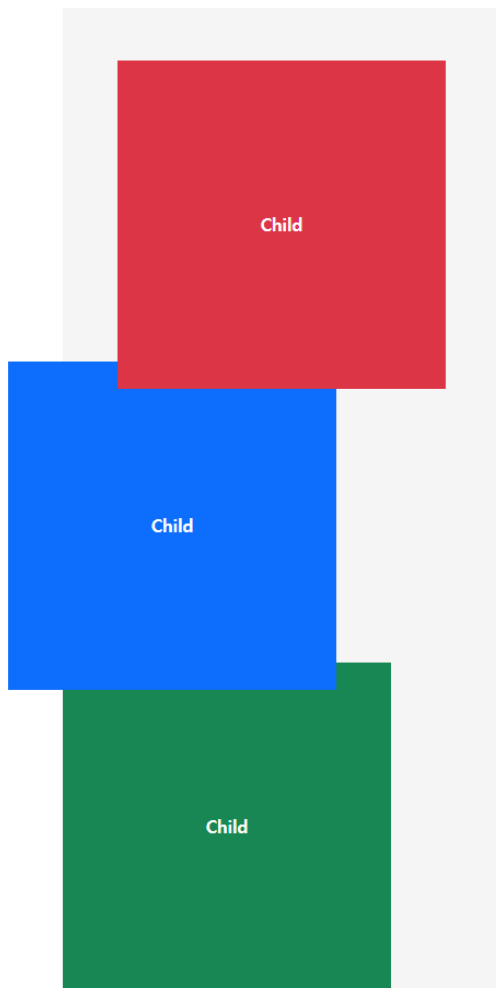
-25

CSS

position

z-index

z-index를 써서 빨강 요소를 파랑보다 앞으로 오게 해보겠습니다.



```
<!DOCTYPE html>
<html>
  <head>...</head>
  <body>
    <div class="parent">
      <div class="child red" style="
        position: relative;
        left: 50px;
        top: 50px;
        z-index: 1;
      ">Child</div> == $0
      <div class="child blue" style="
        position: relative;
        right: 50px;
        top: 25px;
      ">Child</div>
      <div class="child green">Child</div>
    </div>
    <!-- Code injected by live-server -->
    <script type="text/javascript">...</script>
  </body>
</html>
```

html body div.parent div.child.red

스타일 계산됨 레이아웃 이벤트 리스너 DOM 중단점 >>

필터 :hov .cls +

element.style {
 position: relative;
 left: 50px;
 top: 50px;
 z-index: 1;
}

.red {
 background-color: #dc3545;
}

.child {
 width: 300px;
 height: 300px;
 line-height: 300px;
 color: white;
 font-weight: bold;
 text-align: center;
}

* {
 box-sizing: border-box;
 margin: 0;
}

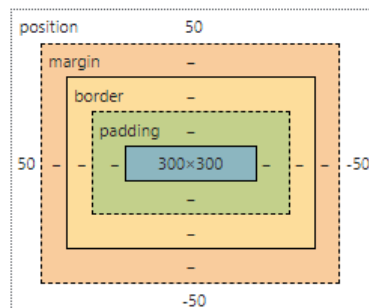
div {
 display: block;
}

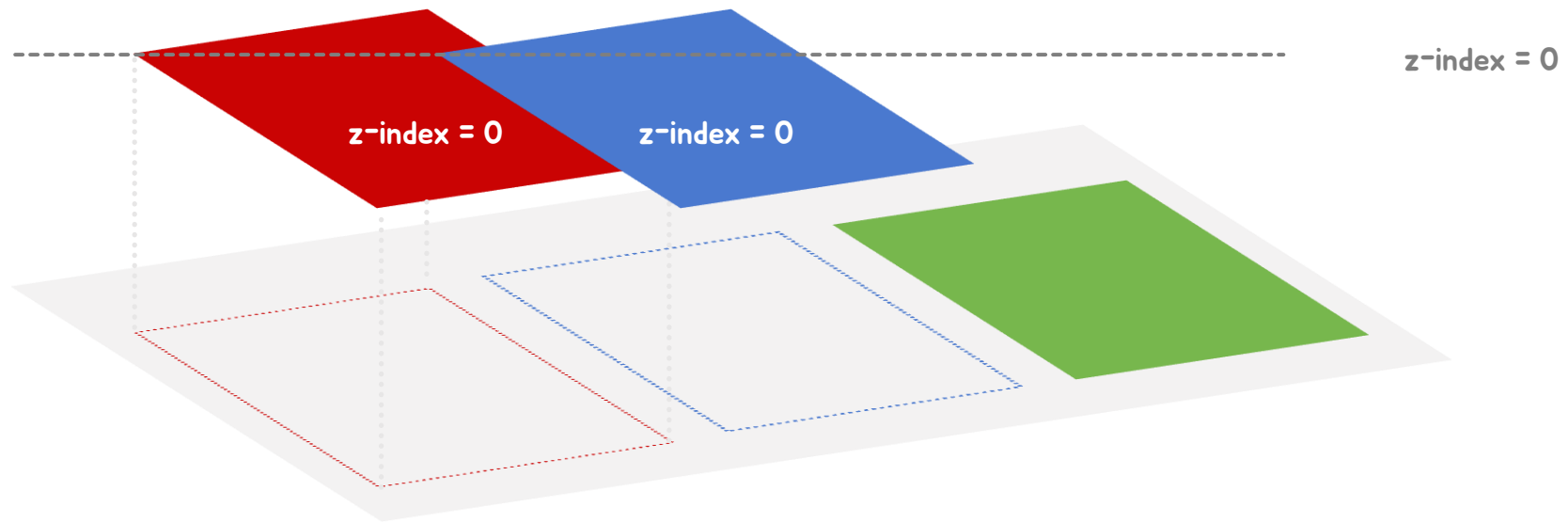
style.css:21

style.css:12

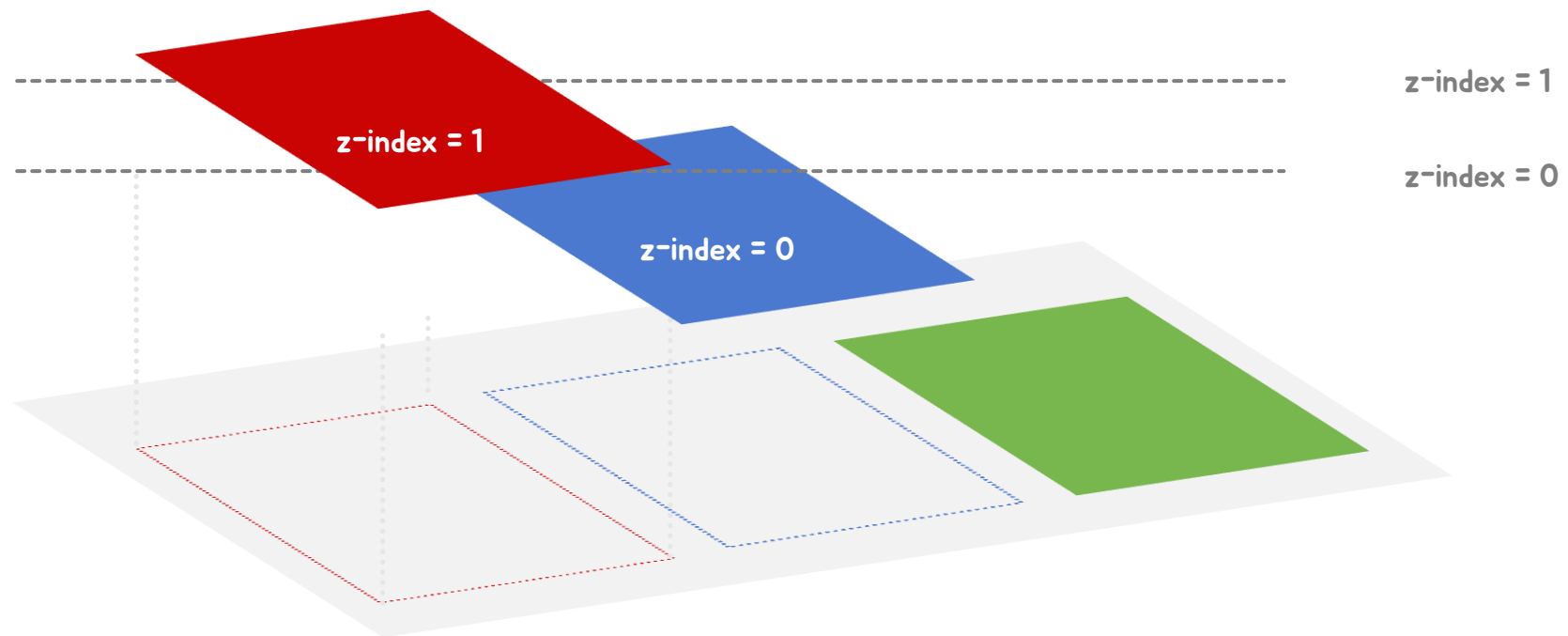
style.css:1

사용자 에이전트 스타일시트





z-index가 크면 클수록 더 위에 놓여지게 됩니다.



CSS

position

static | relative | absolute | fixed | sticky

absolute Float와 유사해짐
기준점을 잡아야 하는데 기준점 대상이 static이면 안됨

fixed Viewport를 기준으로 기준점이 잡혀 있습니다.
내가 보고 있는 브라우저 창의 크기

sticky 스크롤 되는 상위 요소를 기준으로 오프셋을 적용합니다.

top | bottom | right | left

위의 4가지를 선택적으로 위치를 정하여 줍니다.
보편적으로 top과 bottom / right와 left를
동시에 적용하지 않습니다.

z-index

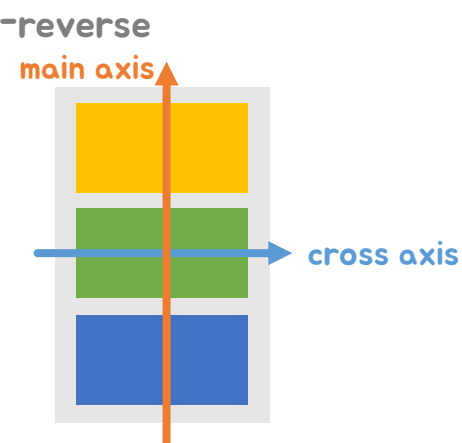
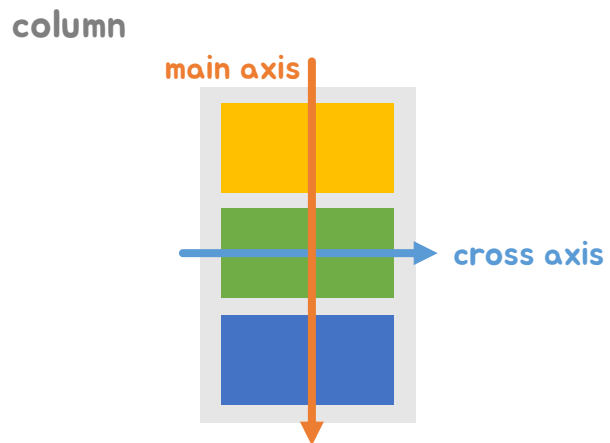
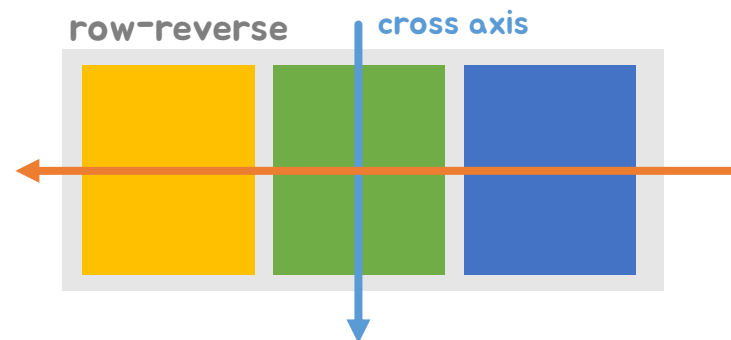
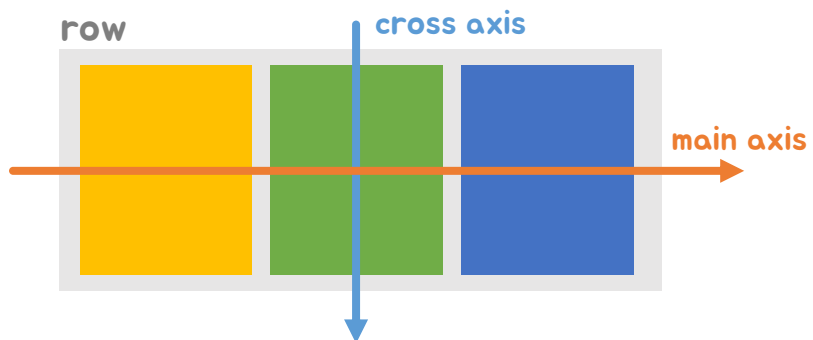
Z축 정도를 나타내서 상위에 무엇이
먼저 나타나야 하는지 값을 지정해줄 수 있습니다.

display: flex

inline-flex

정렬하고자 하는 요소를 감싸는 부모에게 display: flex

flex-direction: row | row-reverse | column | column-reverse





flex-wrap

nowrap | wrap

nowrap: 감싸지 않고 자식의 사이즈를 줄여서라도 한 줄로 정렬

wrap: 한 줄에 모두 정렬하기에 공간이 넉넉하지 않으면 여러 줄을 만듭니다

order 순서를 정해서 바꿔줄 수 있습니다

FLEXBOX FROGGY

◀ 단계 1 of 24 ▶

Flexbox Froggy에 오신 것을 환영합니다! Flexbox Froggy는 Froggy와 친구를 돕는 CSS 코드 게임입니다. 오른쪽의 `justify-content` 속성을 이용하여 개구리가 수련잎으로 이동할 수 있도록 도와주세요. 이 속성은 다음의 값들을 인자로 받아 요소들을 가로선 상에서 정렬합니다:

- `flex-start`: 요소들을 컨테이너의 왼쪽으로 정렬합니다.
- `flex-end`: 요소들을 컨테이너의 오른쪽으로 정렬합니다.
- `center`: 요소들을 컨테이너의 가운데로 정렬합니다.
- `space-between`: 요소들 사이에 동일한 간격을 둡니다.
- `space-around`: 요소들 주위에 동일한 간격을 둡니다.

예를 들어, `justify-content: flex-end;`는 개구리를 오른쪽으로 이동시킵니다.

```
1 #pond {
2   display: flex;
3   
4 }
```

다음

Flexbox Froggy는 다음에 의해 개발되었습니다 [Codepip](#) • [GitHub](#) • [Twitter](#) • [Settings](#)

Want to learn CSS grid? Play [Grid Garden](#).



CSS media

```
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
```

```
@media screen and (min-width: 425px) {  
  .media-test::after {  
    content: "425px";  
  }  
}  
  
@media screen and (min-width: 768px) {  
  .media-test::after {  
    content: "768px";  
  }  
}  
  
@media screen and (min-width: 1024px) {  
  .media-test::after {  
    content: "1024px";  
  }  
}
```