5주차 CSS

상태	승인
설명	CSS 속성 (position,background,flex)

`position`

요소의 위치 지정 방법의 유형(기준)을 설정하는 것

static : 유형없음/배치 불가능

relative : 요소 자신을 기준으로 배치

absolute : 위치 상 부모 요소를 기준으로 배치

fixed : 브라우저(뷰포트)를 기준으로 배치

sticky : 스크롤 영역 기준으로 배치

top : 요소의 position 기준에 맞는 위쪽에서의 거리 설정

auto : 브라우저 계산

단위 : px,em,cm 등 단위

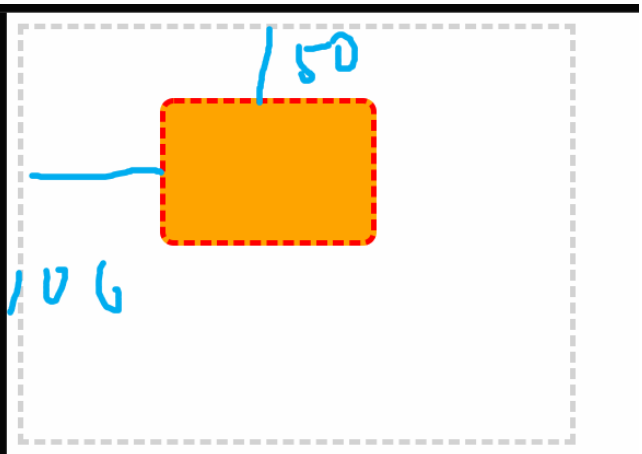
% : 부모요소의 세로너비의 비율로 지정, 음수값 허용

HTML

```html
<div class="parent">
  <div class="child"></div>
</div>
```

CSS

```css
.parent{
  width : 400px;
  height : 300px;
  border : 4px dashed lightgray;
  position : relative;
}

.child{
  width : 150px;
  height: 100px;
  background : orange;
  border : 4px dashed red;
  border-radius : 10px;
  position : absolute;
  top : 50px;
  left : 100px;

}
```



`bottom` `left` `right` 또한 위에서 설명한것과 같은 값을 가진다.

> 위 요소를 적용해도 실제 위치는 원 지점과 같고, 홀로그램처럼 보이는것만 다르게 보여지는 것이다.

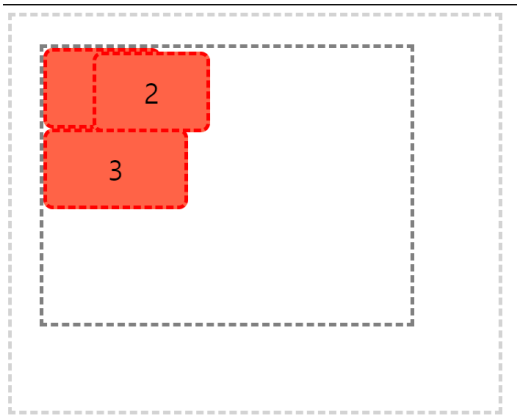
`abolute`
쓰면 붕 떠지는 듯한 느낌
위치상 부모는 `html`상 부모가 아닌, `position = ~~` 으로 설정된 부분의 부모요소를 의미한다.

> 습관적으로 부모요소에 `position`이 있는걸 확인하고 없으면 습관적으로 `relative`를 사용할 것

`fixed`
쇼핑몰의 배너같은 느낌으로 화면에 고정되게 된다.

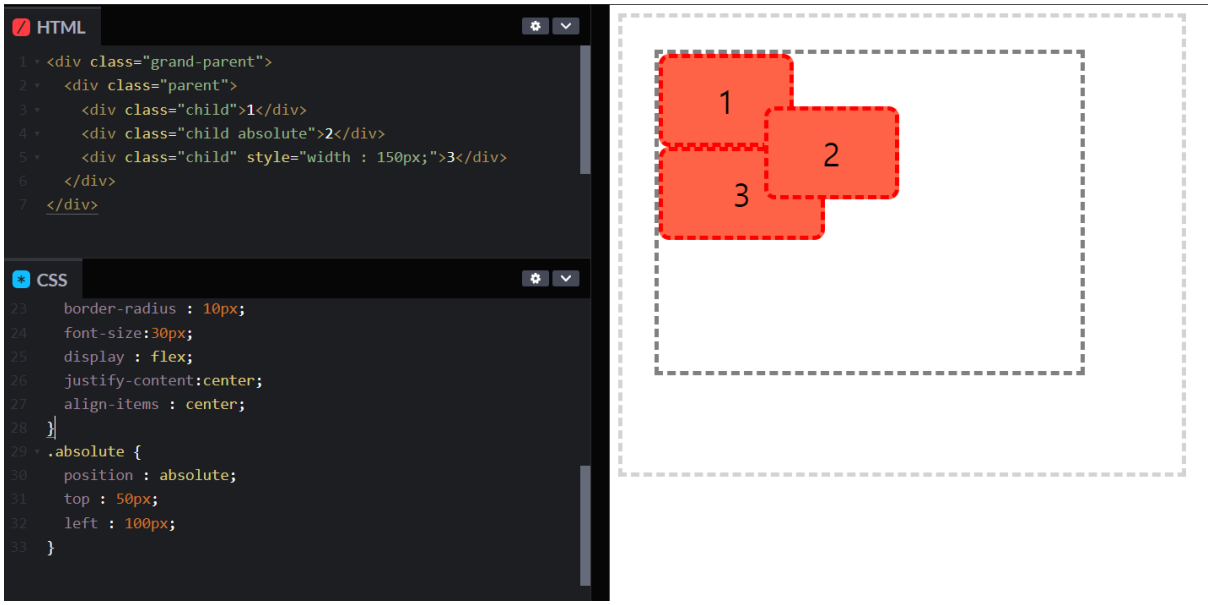
```
<div class="grand-parent">
  <div class="parent">
    <div class="child">1</div>
    <div class="child absolute">2</div>
    <div class="child" style="width : 150px;">3</div>
  </div>
</div>
```

```
body{
  height : 4000px;
}

.grand-parent {
  width : 400px;
  height : 300px;
  padding : 30px 100px 100px 30px;
  border : 4px dashed lightgray;
}

.parent{
  width : 400px;
  height : 300px;
  border : 4px dashed gray;
  position : relative;
}
.child{
  width : 120px;
  height:80px;
  background : tomato;
  border : 4px dashed red;
  border-radius : 10px;
  font-size:30px;
  display : flex;
  justify-content:center;
  align-items : center;
}
.absolute {
  /* 해당 부분의 position이나 top,left값을 수정하면서 실습*/
  position : fixed;
  top : 50px;
  left : 100px;
}
```



위 상태에서 스크롤을 내려보고 2번 박스의 위치가 어떻게 변하는지 생각해 보자.

이번에는 `absolute`로 아래와 같은 배치를 만들어보자.



마찬가지로 scroll을 내려보면서 어떠한 방식으로 2번 박스가 변하는지 알아보자.

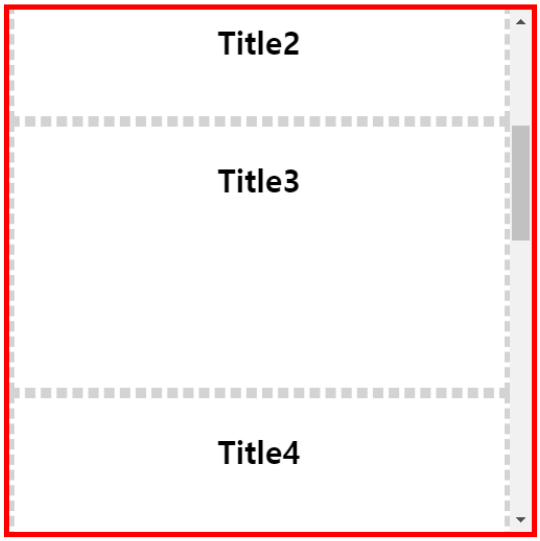
`sticky`

스크롤 영역 기준으로 배치

```
<div class="container">
  <div class="section">
  <h1>Title1</h1>
</div>
```

```
<div class="section">
  <h1>Title2</h1>
</div>
<div class="section">
  <h1>Title3</h1>
</div>
<div class="section">
  <h1>Title4</h1>
</div>
<div class="section">
  <h1>Title5</h1>
</div>
<div class="section">
  <h1>Title6</h1>
</div>
<div class="section">
  <h1>Title7</h1>
</div>
<div class="section">
  <h1>Title8</h1>
</div>
</div>
```

```
.container{
  width : 400px;
  height : 400px;
  border : 4px solid red;
  overflow : auto;
  margin : 50px;
}

.section {
  height : 200px;
  border : 4px dashed lightgray;
}

.section h1{
  text-align:center;
  line-height : 2;
  font-size : 24px;
  font-weight: bold;
  position : sticky;
  top : 0;
}
```



`sticky`를 활용하면 스크롤을 통해 요소가 화면 위에 표시되는 동안만 `fixed`와 동일하게 동작하게 된다.

요소 쌓임 순서(Stack order)
요소가 쌓여있는 순서를 통해 어떤 요소가 사용자와 가깝게 있는지 결정하는 요소

1️⃣ `static`을 제외한 `position`속성의 값이 있을 겨우 가장 위에 쌓임

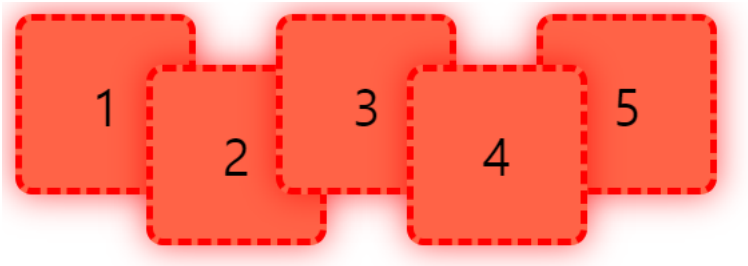
2️⃣ `position`이 모두 존재한다면, `z-index` 속성의 숫자 값이 높을수록 위에 쌓임

3️⃣ `position`속성의 값이 있고, `z-index` 속성 숫자 값이 같다면, `HTML`의 마지막 코드일수록 위에 쌓임(나중에 작성한 코드(요소))

`z-index`는 `position`이 있는 곳에서만 가능
`z-index`를 사용하여 순서설정이 가능하다.

```
<div class="box-group">
  <div class="box box1">1</div>
  <div class="box box2">2</div>
  <div class="box box3">3</div>
  <div class="box box4">4</div>
  <div class="box box5">5</div>
</div>
```

```
.box-group {
  display : flex;
}
.box-group .box{
  width : 100px;
  height: 100px;
  background : tomato;
  border : 4px dashed red;
  border-radius : 10px;
  font-size : 30px;
  display : flex;
  justify-content: center;
  align-items:center;
  margin-right:-30px;
  margin-right : -30px;
  box-shadow: 0 0 20px rgba(255,0,0,.7);
}

.box-group .box:nth-child(2n) {
  margin-top : 30px;
}
/* 각 z-index들을 바꾸어보면서 테스트해보자.*/
.box1 {
  position : relative;

}
.box2 {
  position : relative;
}
.box3 {
  position : relative;
  z-index : 1;

}
.box4 {
  position : relative; /*동시면 html상*/
  z-index : 1;
}
.box5 {
  position : relative;
}
```



`background`
요소의 배경을 설정하는 속성으로 아래와 같은 설정을 가진다.

```
background-color : 배경색상
background-image : 하나 이상의 배경 이미지
background-repeat : 배경이미지의 반복
background-position : 배경이미지의 위치
background-attachment : 배경이미지의 스크롤 여부

background : 색상 이미지경로 반복 위치 스크롤특성;
위와 같은 방식으로 적혀지게 되고, 누락하여 설정해도 괜찮은것이 특징

background color
요소의 배경 색상을 지정
transparent : 투명 이 기본값으로 부여되어 있다.
```

`background-image`
요소의 배경에 하나 이상의 이미지를 삽입하는 방법이다. 이를 활용하면 `img` 태그가 아니더라도 요소에 이미지를 삽입하는 것이 가능하다.

none : 이미지 없음 (기본값)

url("경로") : 이미지 경로

이미지를 여러개 넣는것도 가능하다

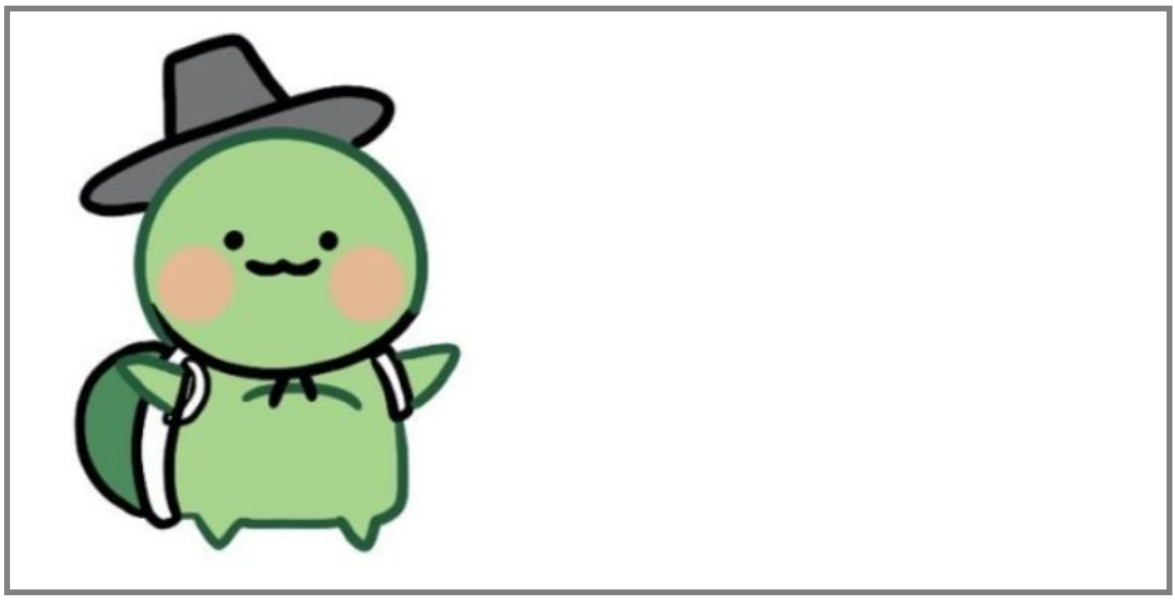
background : url("") no-repeat, url("") no-repeat 100px 50px, ... 이런방식이나

background-image : url("") , url("") 이런 방식의 개별방식으로 두개가 가능하다

background-image는 먼저 삽입한 url이 먼저 올라오는 특징이 있다.

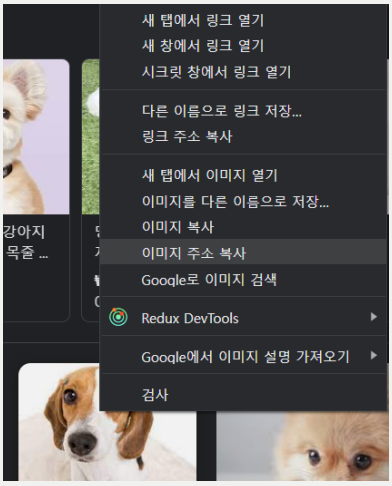
```
<div></div>
```

```
div{
  width : 800px;
  height: 400px;
  border : 4px solid gray;
  margin : 50px;
  overflow:auto;
  background-image: url("https://search.pstatic.net/common/?src=http%3A%2F%2Fblogfiles.naver.net%2FMjAyMTA5MTVfMTM2%2FMDAxNjMxNjgwOTIy
  background-repeat : no-repeat;
  background-size : contain;


}
```



💡 이미지 가져오는법 ❓

원하는 이미지에서 우클릭을 한 후 '이미지 주소 복사' 를 클릭하면 해당 이미지의 url 주소를 가져올 수 있다. 이를 background 의 url 에 넣는다면 이미지를 넣을 수 있다.

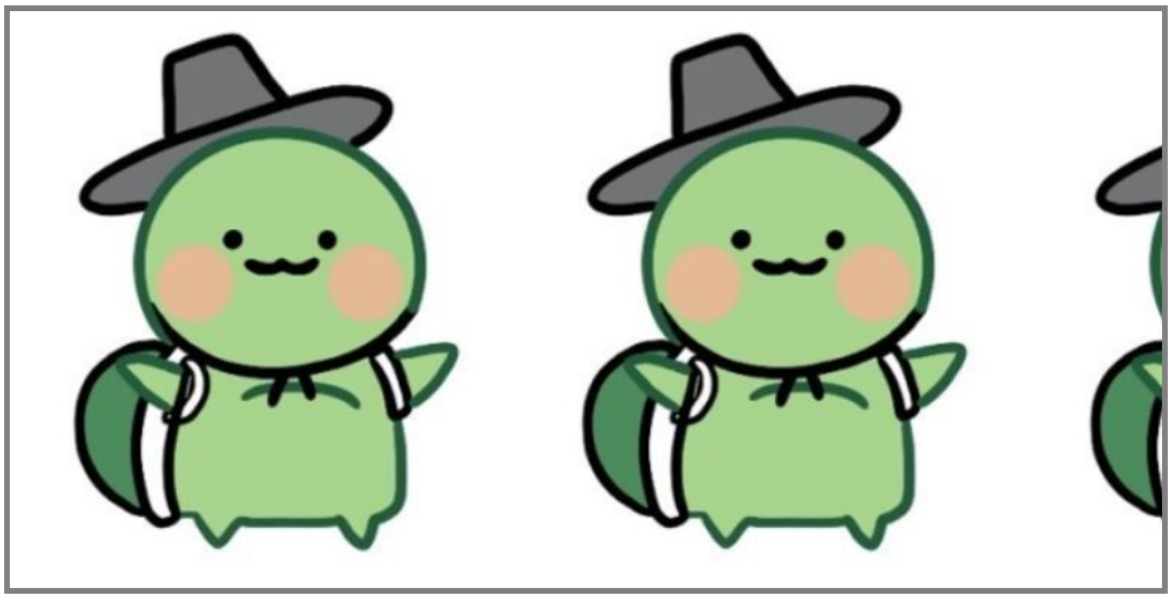


background-image는 먼저 삽입한 url이 먼저 올라오는 특징이 있다.

`background-repeat`
배경 속성의 반복

repeat : 배경이미지를 수직,수평으로 반복
repeat-x : 배경이미지를 수평으로 반복
repeat-y : 배경이미지를 수직으로 반복
no-repeat : 반복없음

```
div{
  width : 800px;
  height: 400px;
  border : 4px solid gray;
  margin : 50px;
  overflow:auto;
  background-image: url("https://search.pstatic.net/common/?src=http%3A%2F%2Fblogfiles.naver.net%2FMjAyMTA5MTVfMTM2%2FMDAxNjMxNjgwOTIy
  background-repeat : repeat-x;
  background-size : contain;


}
```



`background-position`

`%`: 왼쪽상단모서리는 0% 0% , 오른쪽 하단 모서리는 100% 100%

>> x축 y축 순서대로 입력을 해야한다.

>>특이한것은 100%를 하면 안나와야한다고 생각할수 있지만, 실제로는 오른쪽 끝에 붙어서 나오게 된다., 중앙도 마찬가지

`방향`: top,bottom,left,right,center

>> 순서가 상관이 없다

```html
<div></div>
```

```css
div{
  background-image:
url("https://search.pstatic.net/common/?
src=http%3A%2F%2Fblogfiles.naver.net%2FMjAyMT
A5MTVfMTM2%2FMDAxNjMxNjgwOTIyNDEy.II528ejr-
8GDey93I2JhBVxc2wF_YG3Pp8ZWiplkiq4g.TQSNfpOZv
61-D6dRlk_u8s_kPw-
EgY1uqsYMImT0rXgg.JPEG.gsurc%2F%25BA%25ED20.J
PG&type=sc960_832");
  background-repeat : no-repeat;
  width : 600px;
  height : 500px;
  border : 2px dashed lightgray;
  background-position : right bottom;

}
```

단위 : `px` `em` `cm` 등의 단위

>> x축 y축 순서대로 입력을 해야한다.

>> 왼쪽과 오른쪽에서부터의 거리만 계산이 가능한것이 특징

방향과 단위를 동시에 쓰는것도 가능하지만, 순서가 뒤바뀌면 되지 않으므로 순서를 잘 생각하고 써야한다.

아래는 방향과 단위를 동시에 사용한 예시이다.

```css
div{
  background-image:
url("https://search.pstatic.net/common/?
src=http%3A%2F%2Fblogfiles.naver.net%2FMjAyMT
A5MTVfMTM2%2FMDAxNjMxNjgwOTIyNDEy.II528ejr-
8GDey93I2JhBVxc2wF_YG3Pp8ZWiplkiq4g.TQSNfpOZv
61-D6dRlk_u8s_kPw-
EgY1uqsYMImT0rXgg.JPEG.gsurc%2F%25BA%25ED20.J
PG&type=sc960_832");
  background-repeat : no-repeat;
  width : 600px;
  height : 500px;
  border : 2px dashed lightgray;
  background-position : 50px bottom;

}
```

`background-size`

배경이미지의 크기를 지정

`auto` : 배경이미지가 원래의 크기로 표시

단위 : px,em,% 등

`cover` : 배경이미지가 크기 비율을 유지하며, 요소의 더 넓은 너비에 맞춰진다

>>이미지가 잘릴 수 있음

```
<div class="container">
  <div class="for-scroll"></div>
</div>
```

```
.container{
  width : 800px;
  height: 400px;
  border : 4px solid gray;
  margin : 50px;
  overflow:auto;
  background-image: url("https://search.pstatic.net/common/?src=http%3A%2F%2Fblogfiles.naver.net%2FMjAyMTA5MTVfMTM2%2FMDAxNjMxNjgwOTIy
  background-repeat : no-repeat;
  background-size : 200px;


}
```

`size`를 한개만 사용하게 되면 아래와 같이 가로를 기준으로 세로는 자동으로 정해진다.

`cover`를 사용한 예제

HTML

```html
<div class="container">
  <div class="for-scroll"></div>
</div>
```

CSS

```css
.container{
  width : 800px;
  height: 400px;
  border : 4px solid gray;
  margin : 50px;
  overflow:auto;
  background-image:
url("https://search.pstatic.net/common/?
src=http%3A%2F%2Fblogfiles.naver.net%2FMjAyMT
A5MTVfMTM2%2FMDAxNjMxNjgwOTIyNDEy.II528ejr-
8GDey93I2JhBVxc2wF_YG3Pp8ZWiplkiq4g.TQSNfpOZv
61-D6dRlk_u8s_kPw-
EgY1uqsYMImT0rXgg.JPEG.gsurc%2F%25BA%25ED20.J
PG&type=sc960_832");
  background-repeat : no-repeat;
  background-size : cover;


}
```

`contain`

배경이미지가 크기 비율을 유지하며, 요소의 더 짧은 너비에 맞춰짐

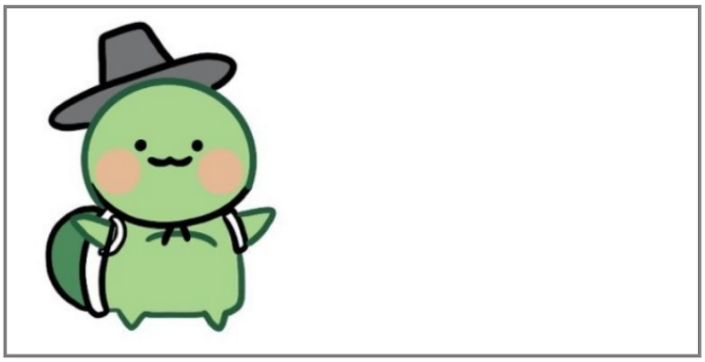
⏩ 이미지가 잘리지 않음 , 요소의 빈공간이 보일 수 있음.

HTML

```html
<div class="container">
  <div class="for-scroll"></div>
</div>
```

CSS

```css
.container{
  width : 800px;
  height: 400px;
  border : 4px solid gray;
  margin : 50px;
  overflow:auto;
  background-image:
url("https://search.pstatic.net/common/?
src=http%3A%2F%2Fblogfiles.naver.net%2FMjAyMT
A5MTVfMTM2%2FMDAxNjMxNjgwOTIyNDEy.II528ejr-
8GDey93I2JhBVxc2wF_YG3Pp8ZWiplkiq4g.TQSNfpOZv
61-D6dRlk_u8s_kPw-
EgY1uqsYMImT0rXgg.JPEG.gsurc%2F%25BA%25ED20.J
PG&type=sc960_832");
  background-repeat : no-repeat;
  background-size : contain;


}
```



`animation`

요소에 애니메이션을 설정/제어

animation-name @keyframes규칙의 이름을 지정

animation-duration 애니메이션의 지속 시간 설정

animation-timing-function 타이밍 함수 지정

animation-delay 애니메이션의 대기 시간 설정

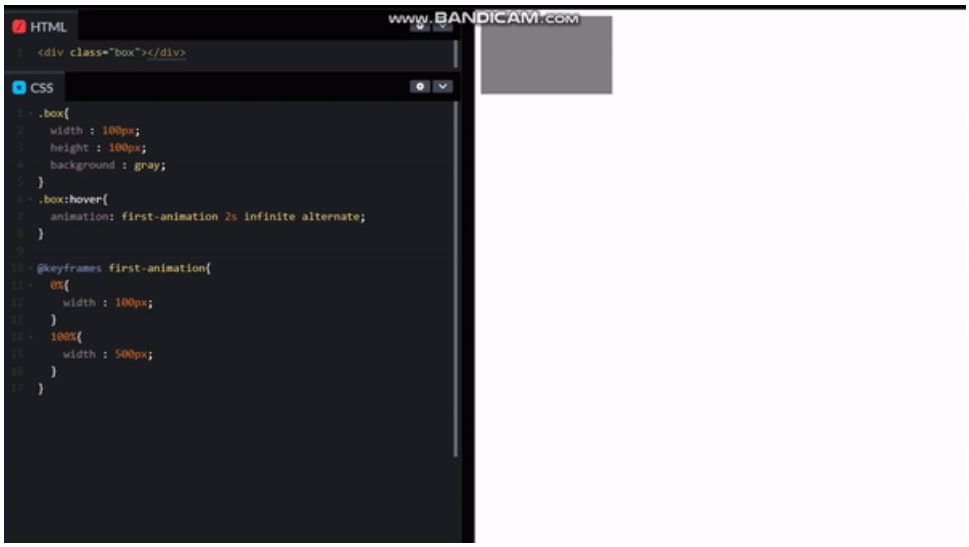
animation-iteration-count 애니메이션의 반복 횟수 설정

animation-direction 애니메이션의 반복 방향 설정

animation-fill-mode 애니메이션의 전후 상태 설정

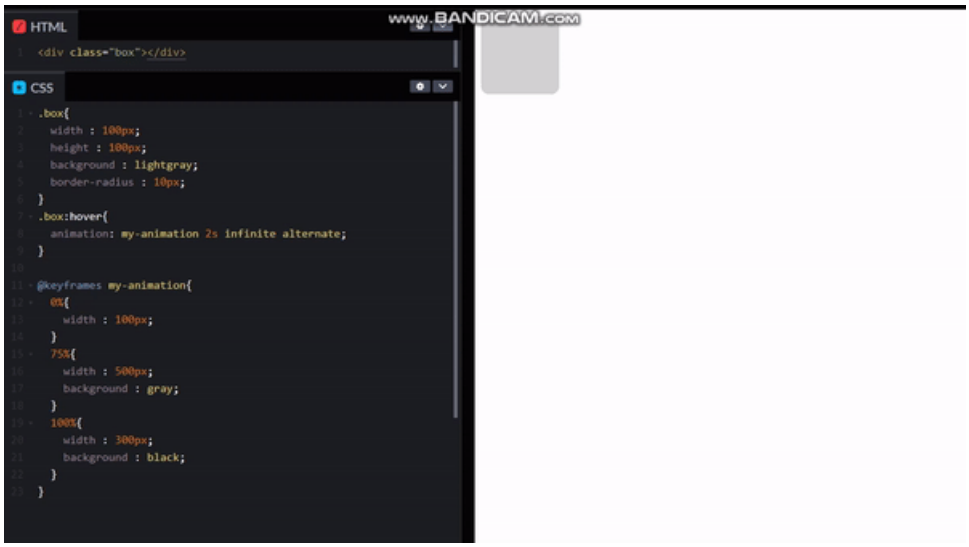
animation-play-state 애니메이션의 재생과 정지 설정

>>애니메이션을 직접 만드는건 아니고, 특정 keyframes에 의해 진행된다.



`@keyframes`

2개 이상의 애니메이션 중간 상태(프레임)을 지정



`animation-name`

none : 애니메이션을 지정하지 않음

@keyframes 이름 : 이름이 일치하는 @keyframes 규칙의 애니메이션을 적용

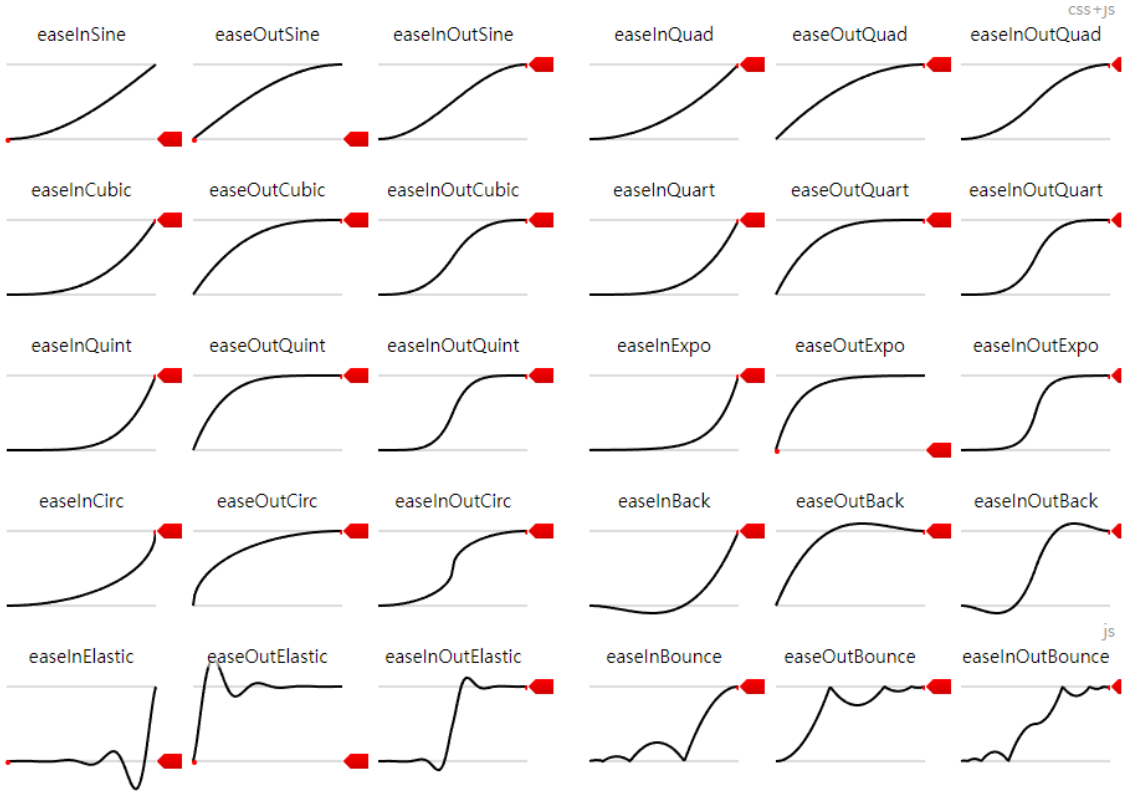
`animation-duration`

기본 : 0s

ms도 사용이 가능하다.

`animation-timing-function`

타이밍함수(애니메이션 효과를 계산하는 방법)지정



`animation-direction`

애니메이션의 반복 방향을 설정

normal : 정방향만 반복

reverse : 역방향만 반복

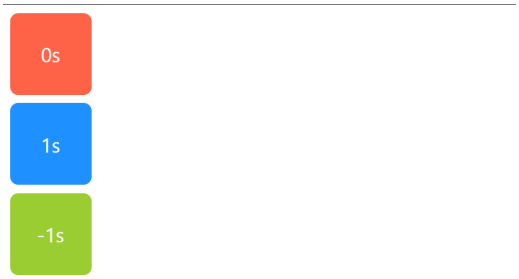
alternate : 정방향에서 역방향으로 왕복

alternate-reverse : 역방향에서 정방향으로 왕복

```
<div class="box box1">0s</div>
<div class="box box2">1s</div>
<div class="box box3">-1s</div>
```

```
.box{
  width : 100px;
  height : 100px;
```

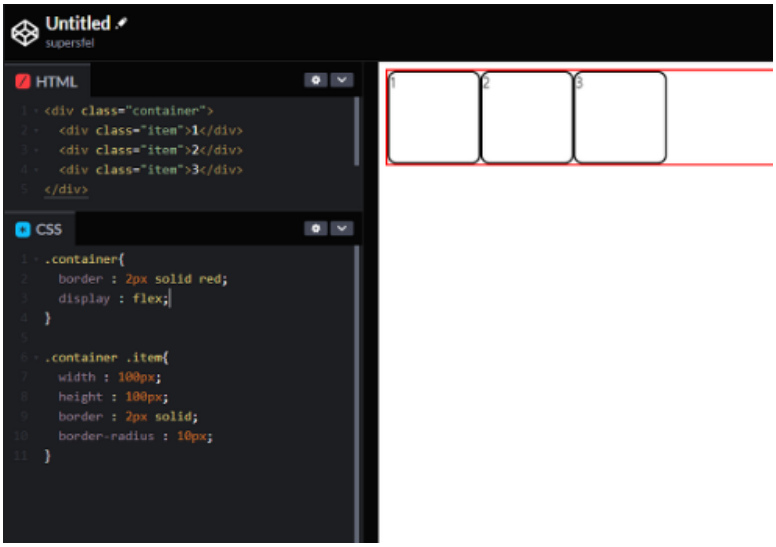
```
  background : lightgray;
  border-radius : 10px;
  margin : 10px;
  color : white;
  font-size : 24px;
  display : flex;
  justify-content : center;
  align-items : center;

}

.box1{background : tomato;}
.box2{background : dodgerblue;}
.box3{background : yellowgreen;}

.box1:hover{
  animation : size-up 1s alternate-reverse;
  animation-timing-function : linear;
  animation-delay : 0s;

}
.box2:hover{
  animation : size-up 1s 2 alternate;
  animation-timing-function : linear;
  animation-delay : 0s;
}
.box3:hover{
  animation : size-up 1s 2 alternate;
  animation-timing-function : linear;
  animation-delay : -1s;
}
.box:hover{

}

@keyframes size-up {
  0%{
    width: 150px;
  }
  100%{
    width : 500px;
  }
}
```



`CSS FLEX(Flexible Box)`

Flex는 요소의 크기가 불분명 하거나, 동적인 경우에도 각 요소를 정렬할 수 있는 효율적인 방법을 제시한다.

`Container`와 `item`으로 구성되어 사용된다.
`Container` : display,flex-flow,justify-content
`item` : order,flex,align-self 등의 속성을 사용할 수 있다.



`Flex Container`

display : `Flex Container`를 정의
flex-flow : `Flex-direction`와 `flex-wrap`의 단축 속성
flex-direction `Flex items`의 주 축을 설정
flex-wrap : `Flex Items`의 여러 줄 묶음 설정
justify-content 주축의 정렬 방법을 설정
align-content 교차 축의 정렬 방법 설정
align-items 교차축에서 `items`의 정렬 방법 설정

`Flex-direction`

row : items를 수평(좌에서 우)으로 표시

row-reverse : row의 반대방향

column : items를 수직축(위에서아래)로 표시

column-reverse : column의 반대 방향

```
<div class="container">
  <div class="item item1"></div>
  <div class="item item2"></div>

</div>
```

```
.container{
  border : 4px solid red;
  display : flex;
  flex-direction : column /*해당부분 실습*/
}
.container .item{
  width : 100px;
  height : 100px;
  background : gray;
  border : 4px dashed black;
  border-radius : 10px;
}
```

`justify-content`

주 축의 정렬 방법을 설정

```
flex-start : 시작점으로 정렬
flex-end : 끝점으로 정렬
center : 중앙으로 정렬
space-between : 첫,마지막아이템은 시작,끝점에 붙이고 나머지는 동일간격
space-around : 모든 아이템당 여백이 좌우가 같게 정렬
```

```
<div class="container">
  <div class="item item1"></div>
  <div class="item item2"></div>

</div>
```

```
.container{
  border : 4px solid red;
  display : flex;
  height : 400px;
  justify-content : space-around; /* 해당부분을 바꿔면서 실습*/
  align-items : center;
}
.container .item{
  width : 100px;
  height : 100px;
  background : gray;
  border : 4px dashed black;
  border-radius : 10px;
}
```

`align-items`
교차축에서 `items`의 정렬 방법을 설정하는 요소

`itmes`가 한 줄일 경우 많이 사용한다.

```
여러중일경우 align-content속성이 우선되게 된다.
stretch : Container의 교차축을 채우기 위해 items을 늘림
flex-start : 시작점으로 정렬
flex-end : 끝점으로 정렬
center : 중앙으로 정렬
baseline : items를 문자 기준선에 정렬
```

위에서 실행했던 예제에서 `align-items`부분을 바꿔가면서 실습해보자.

💡 글씨의 중앙 정렬 ❓

이러한 flex의 성질을 활용하면 요소의 중앙정렬을 쉽게 할 수 있다.

```
<div class="container">
  <div class="item item1">1</div>
  <div class="item item2">2</div>

</div>
```

```
.container{
  border : 4px solid red;
  display : flex;
  height : 400px;
  justify-content : space-around;
  align-items : center;
}
.container .item{
  width : 100px;
  height : 100px;
  background : gray;
  border : 4px dashed black;
  border-radius : 10px;
  /* 중앙정렬 */
  display : flex;
  justify-content  : center;
  align-items : center;

  font-size : 20px;
  color : white;
}
```

💡 문제 1

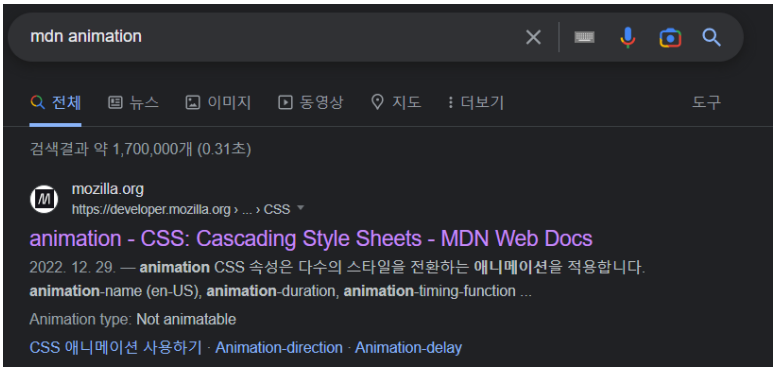
아래 화면을 구현해 보시오

힌트 : flex요소 안의 요소는 어떤 형식의 요소도 될 수 있다.

지금까지 대표적인 `css` 속성들을 알아보았다. 모든 `css` 속성을 알아보지는 않았으며 각 속성또한 모든 기능을 배워보지는 않았다.

`css` 를 들어가기 앞서 설명했듯이 `css` 에는 정말 많은 속성이 존재하고 이에따른 사용방법과 옵션이 존재한다. 따라서 본인이 필요한 `css` 속성을 아래 페이지에서 찾아서 사용하는 능력이 필요하다.

만약, 본 교재에 없는 `css` 속성을 사용하고 싶다면 이전에 소개한 `MDN` 문서를 활용하면 된다. 만약 `animation` 속성에 대해 궁금하다면



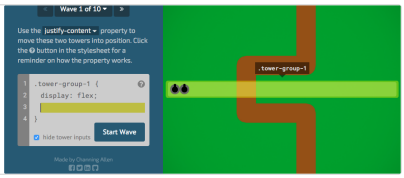
다음과같이 검색하여 MDN 문서에서 animation 속성의 모든 사용방법과 옵션을 확인할 수 있다.

FLEX 게임

Flexbox Defense

Your job is to stop the incoming enemies from getting past your defenses. Unlike other tower defense games, you must position your towers using CSS!

http://www.flexboxdefense.com/



flex의 기능을 게임으로 배우면서 할 수 있다. google에 flexbox-defense로 검색한 후에 단계를 격파해보자

💡 문제 1 ❓

아래 페이지를 만들어보시오 (형식 자유, 이미지는 검색하여 사용해 볼 것)