display: flex

display: flex
Flexible Box Layout

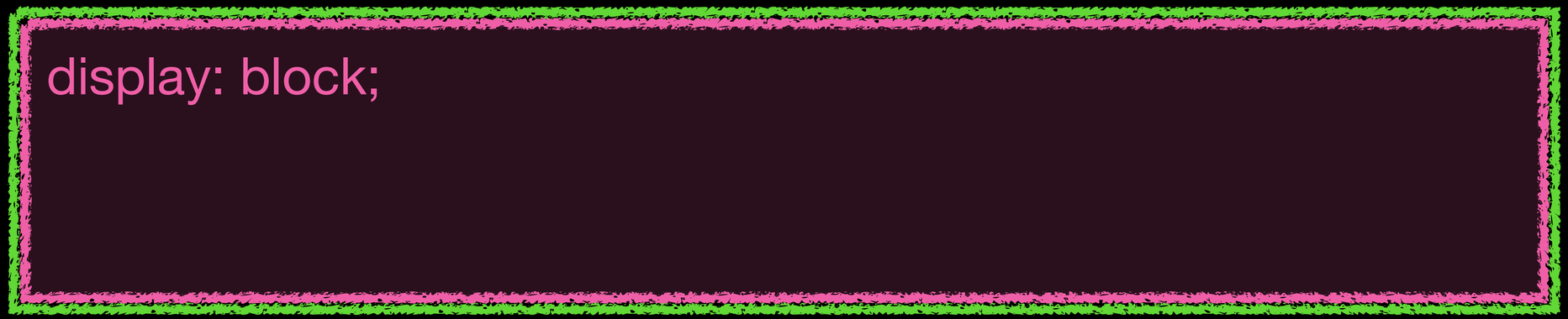
display: flex

Flexible Box Layout

Flexbox

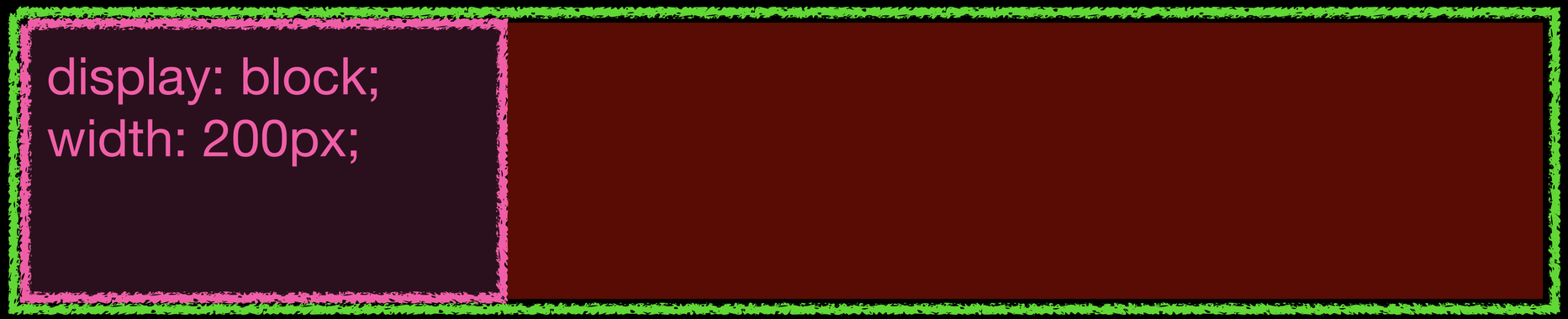
부모 요소와 직계 자식 요소간의
관계를 기반으로 레이아웃을
잡을 수 있게 해주는 속성

Flexbox에서의 **가용 공간**을 여백으로 활용

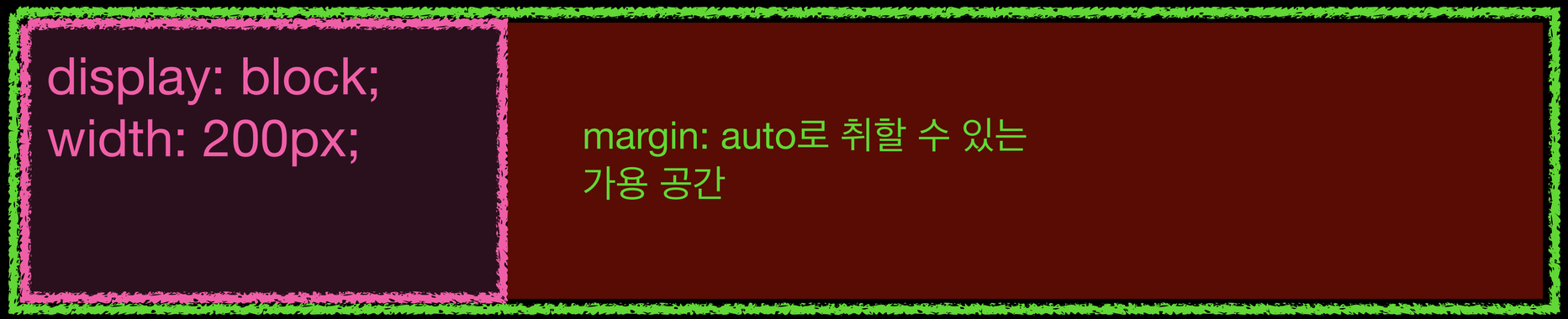


display: block;

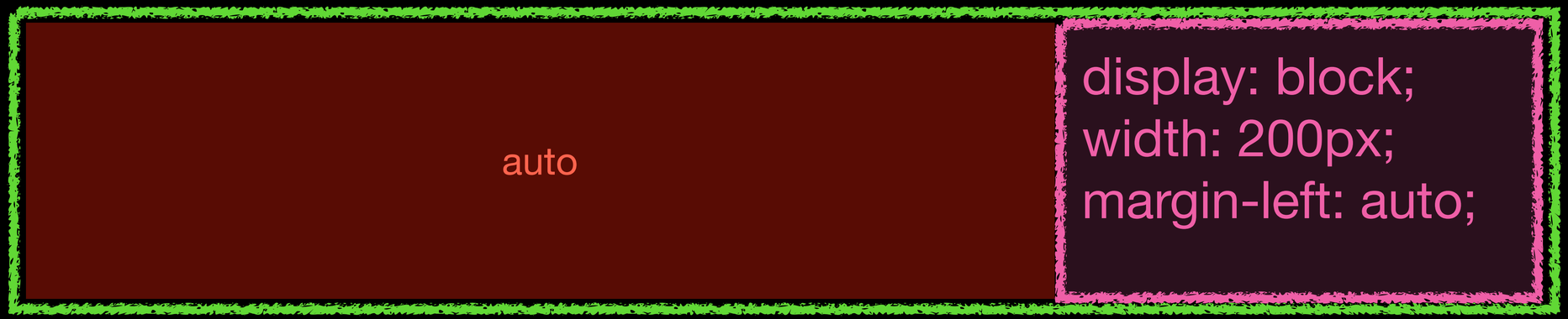
display: block;



display: block;

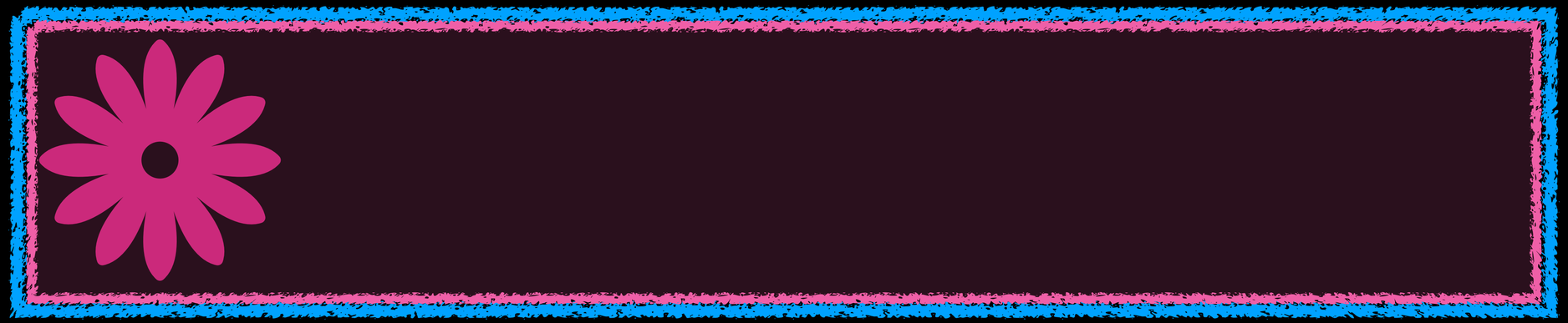


display: block;



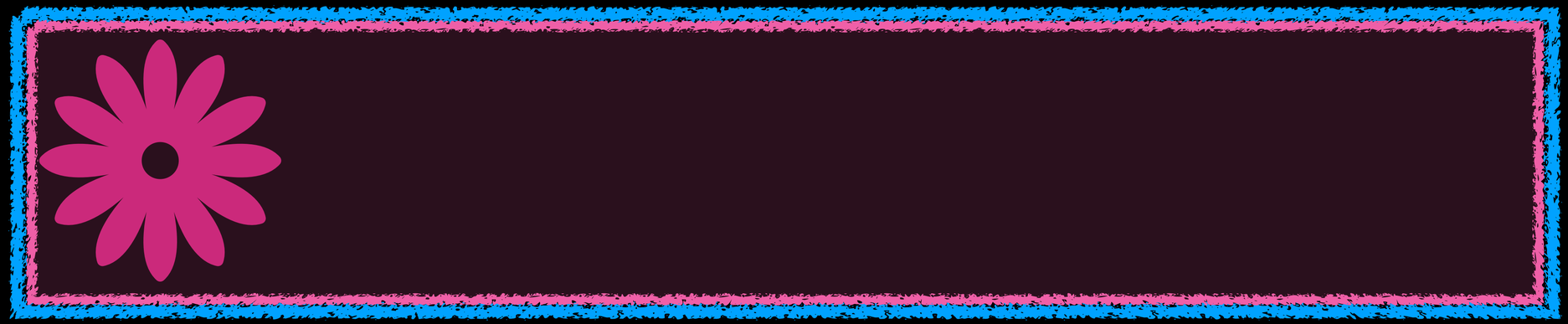
display: block;


auto
display: block;
width: 200px;
margin-left: auto;
margin-right: auto;
auto
display: block;

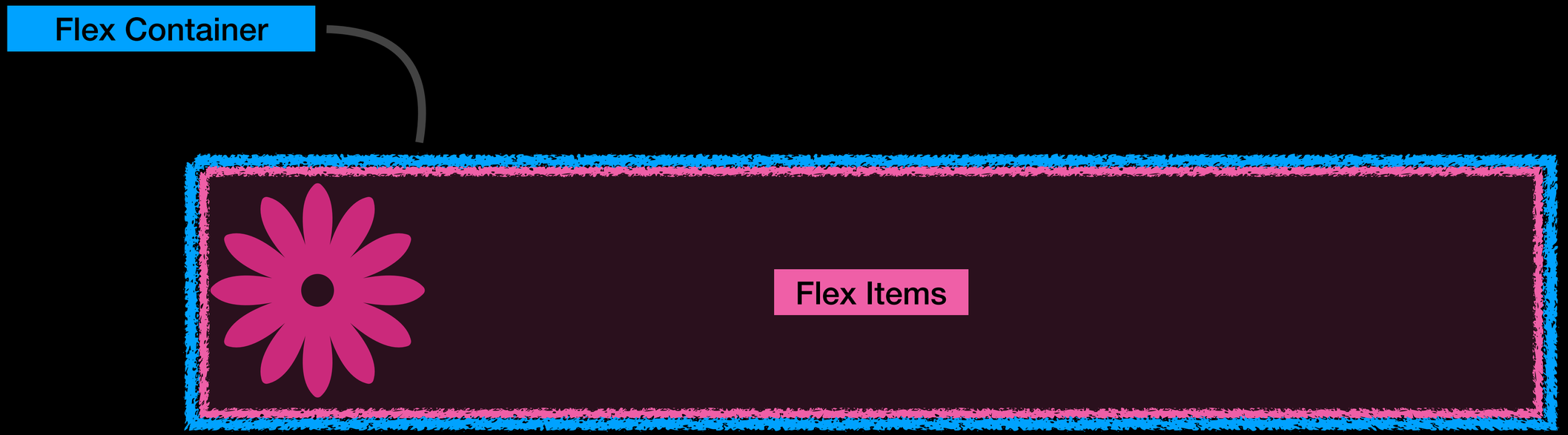


display: flex;

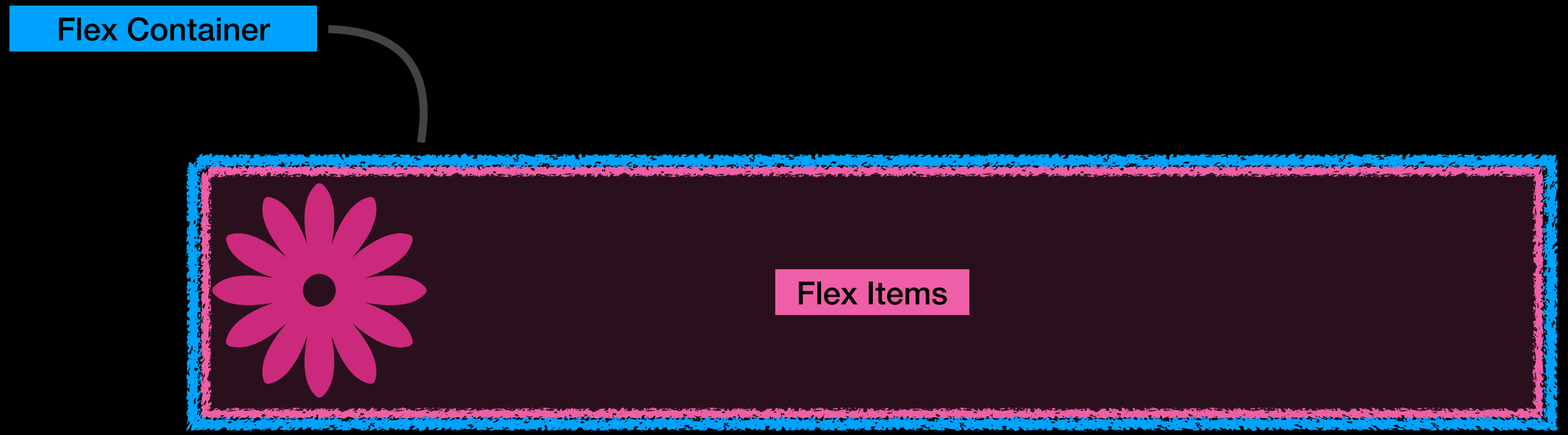
Flex Container



display: flex;



display: flex;

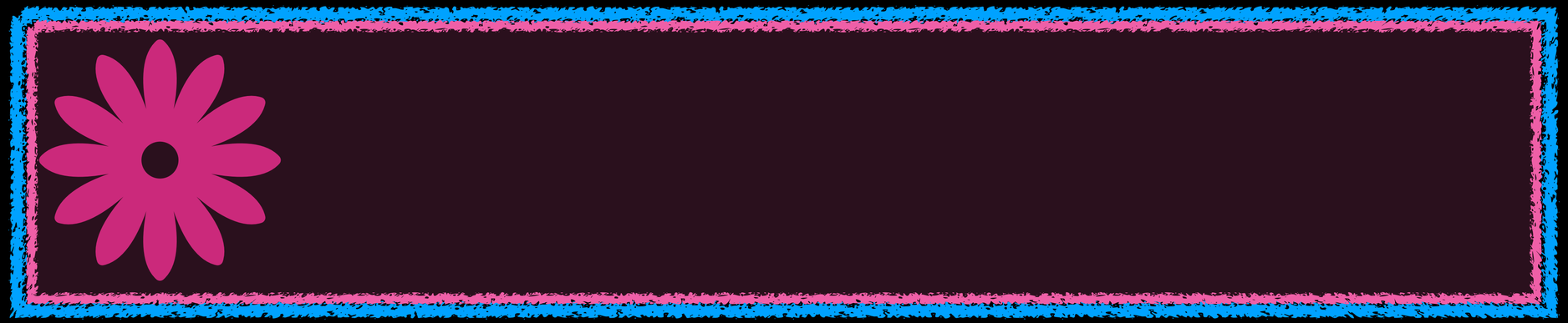


display: flex;

```
<div class="flex-item">
    <img src="flower.png">
</div>
```

Flex Container

Flex Items

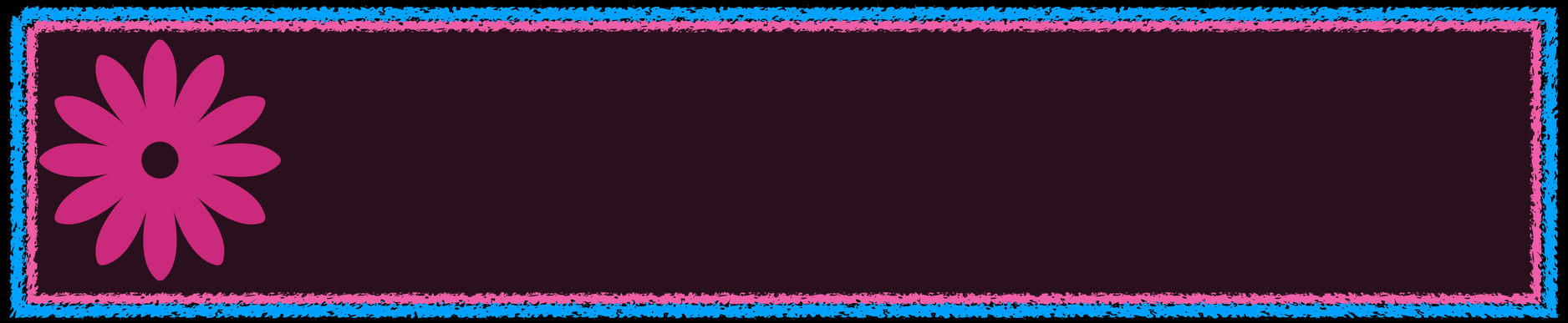


display: flex;

Flex Container

Flex Items

flex-basis: auto에 의해!



display: flex;

Flex Container

Flex Items

flex-basis: auto에 의해! 내부 콘텐츠 크기 만큼으로 조절됨

display: flex;

Flex Container

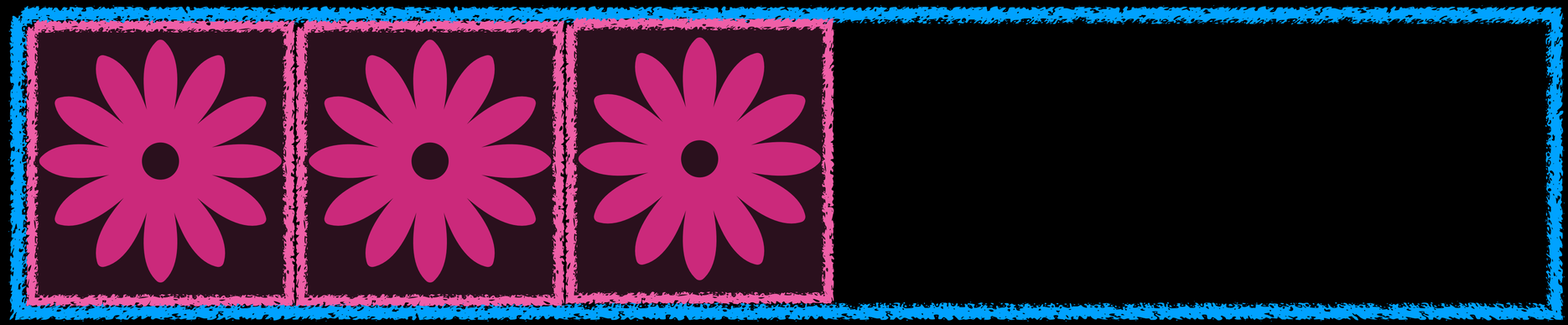
Flex Items

flex-basis: auto에 의해! 내부 콘텐츠 크기 (이미지) 만큼으로 조절됨

display: flex;

Flex Container

Flex Items



display: flex;

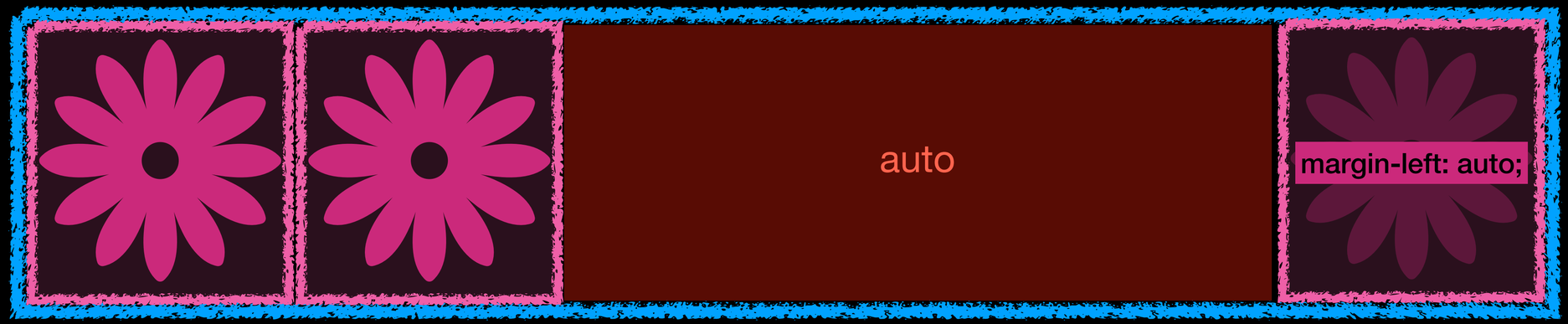
Flex Container

Flex Items

display: flex;

Flex Container

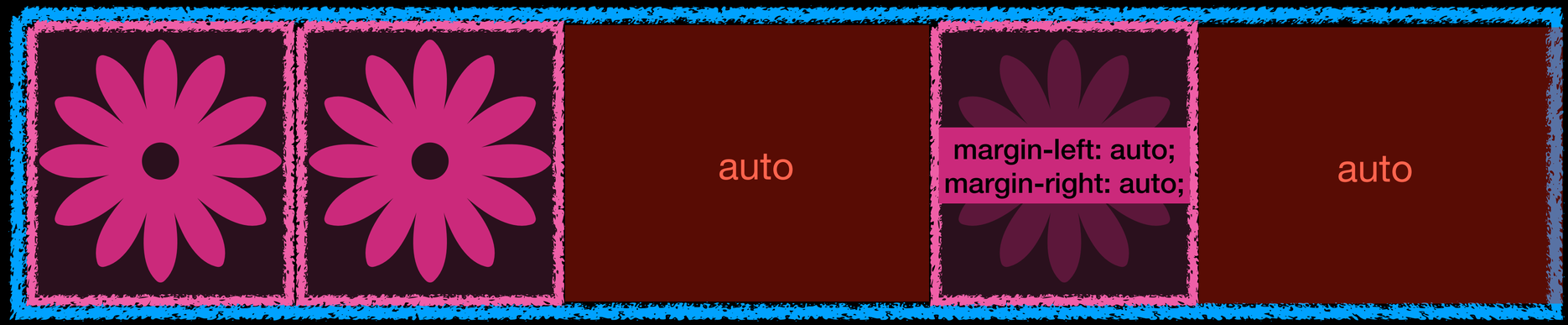
Flex Items



display: flex;

Flex Container

Flex Items



display: flex;

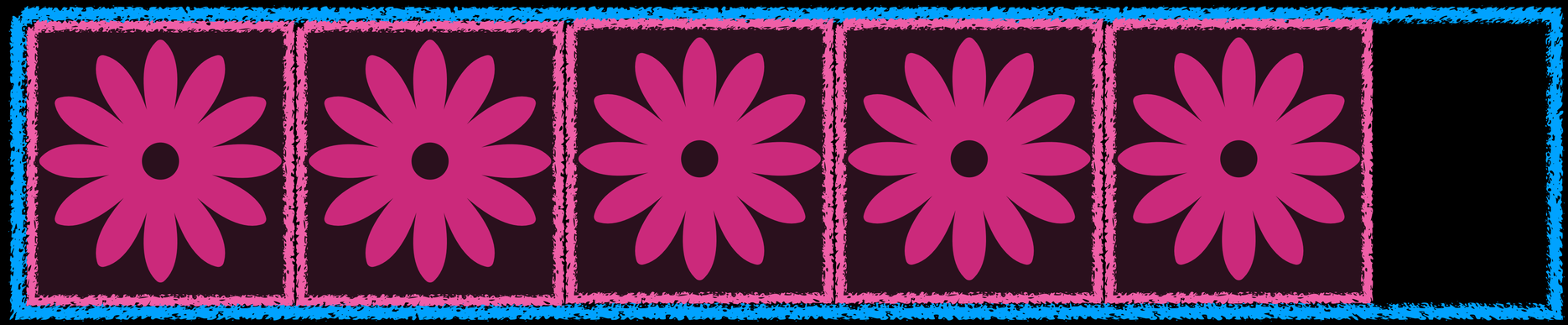
Flex Container

Flex Items

display: flex;

Flex Container

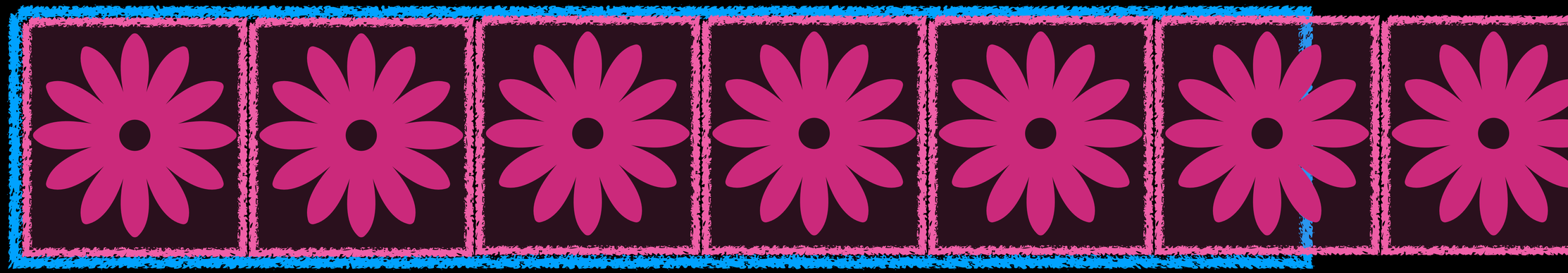
Flex Items



display: flex;

Flex Container

Flex Items

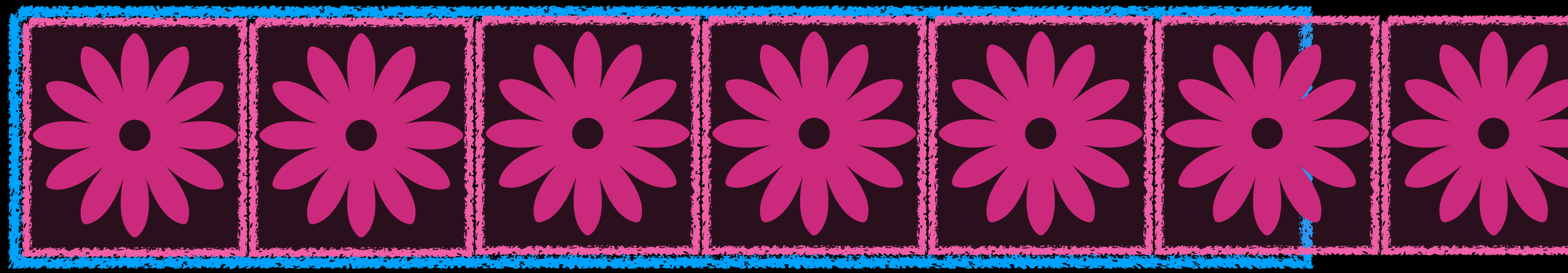


display: flex;

Flex Container

Flex Items

Main Axis



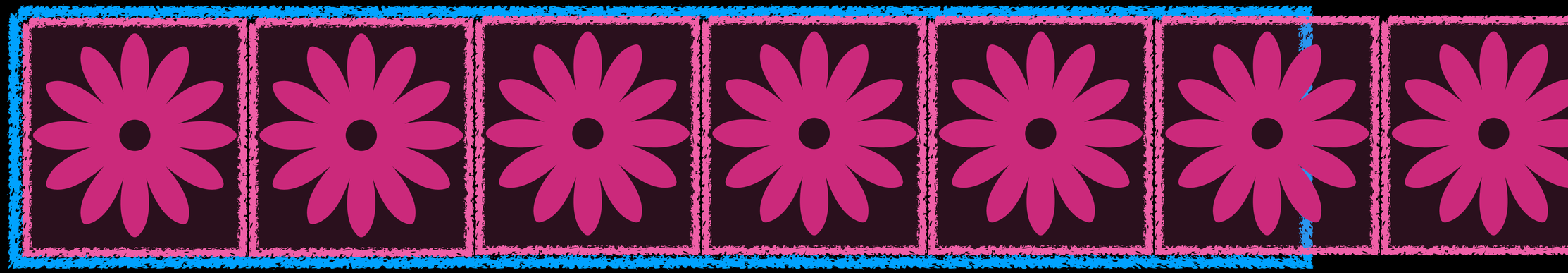
display: flex;

Flex Container

Flex Items

Main Axis

Cross Axis



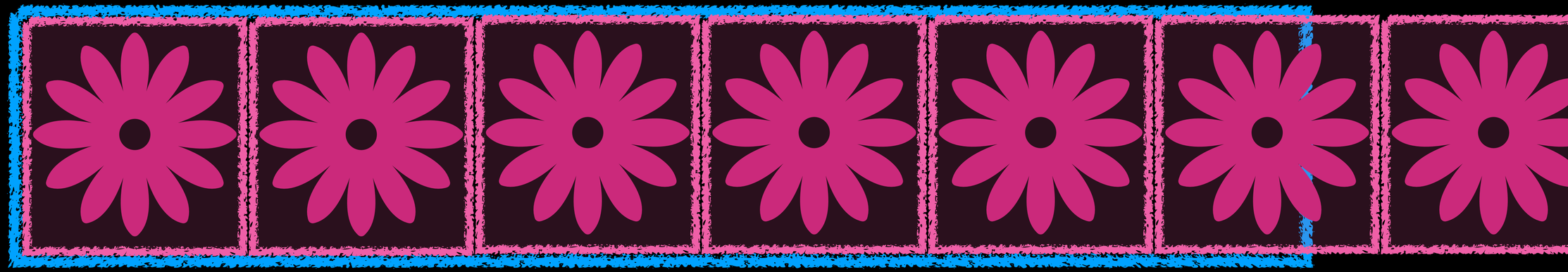
display: flex;

Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용



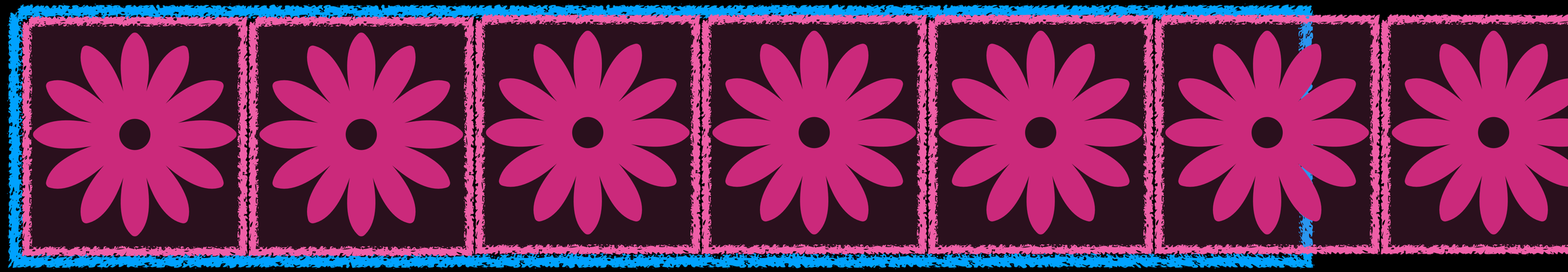
display: flex;

Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용



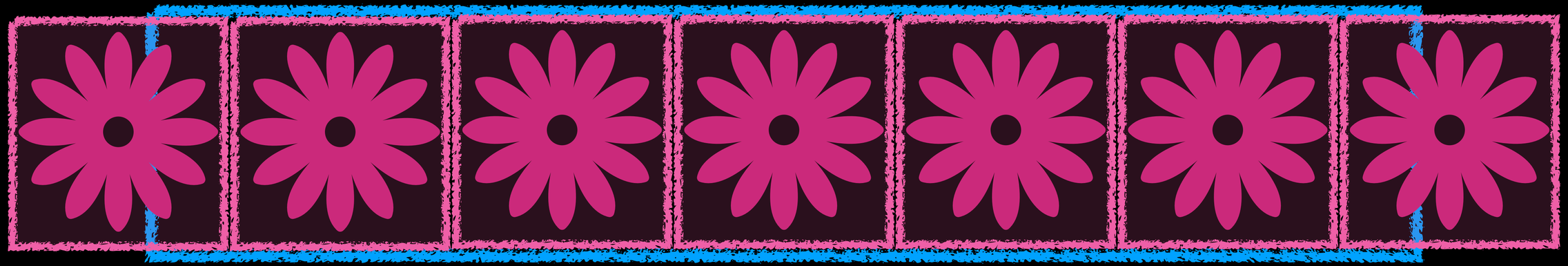
display: flex;
justify-content: center;

Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용



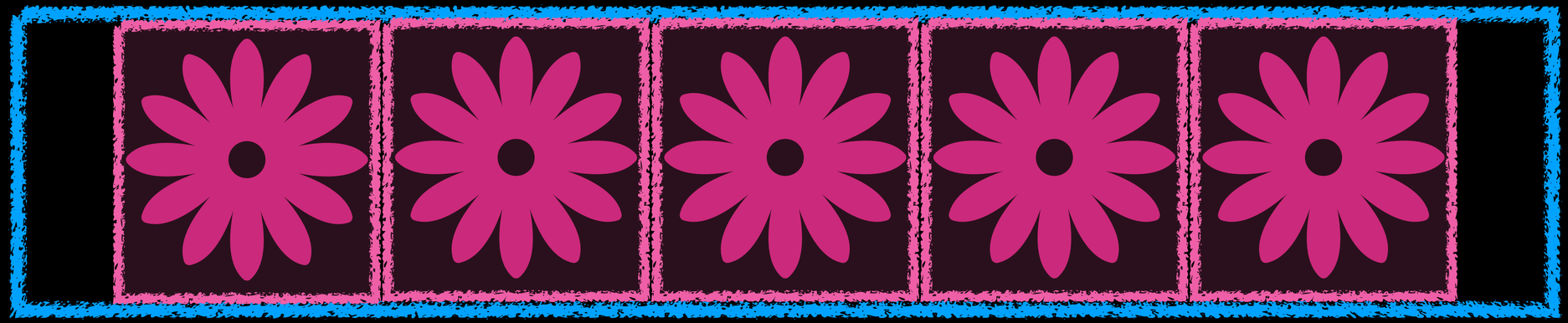
display: flex;
justify-content: center;

Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용



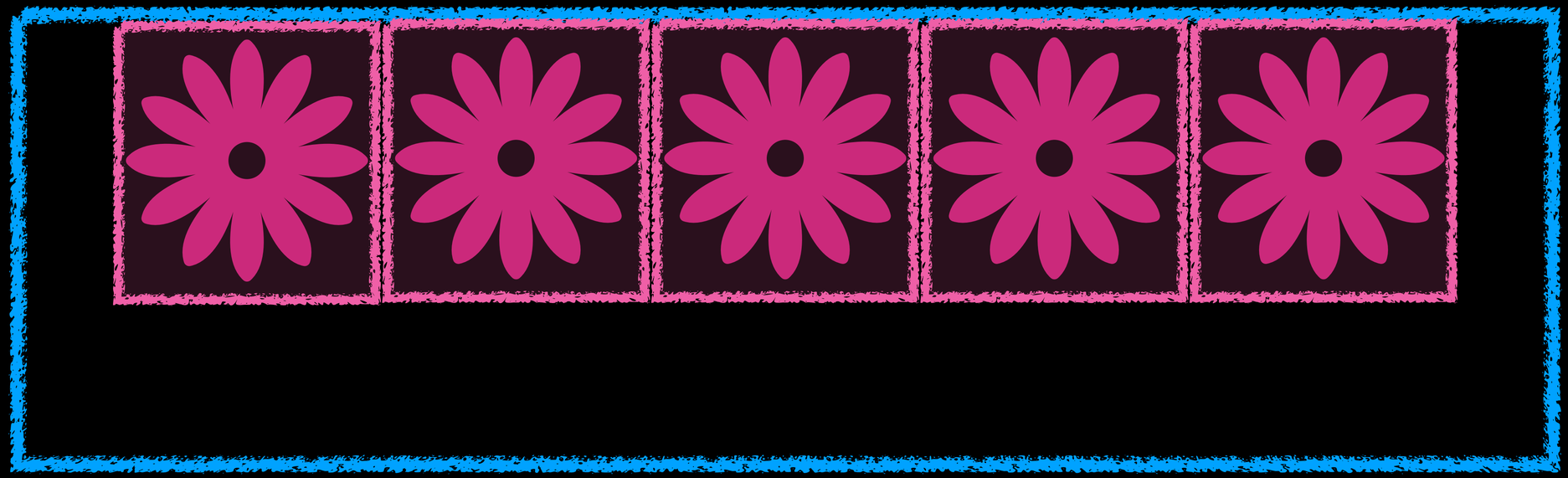
display: flex;
justify-content: center;

Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용



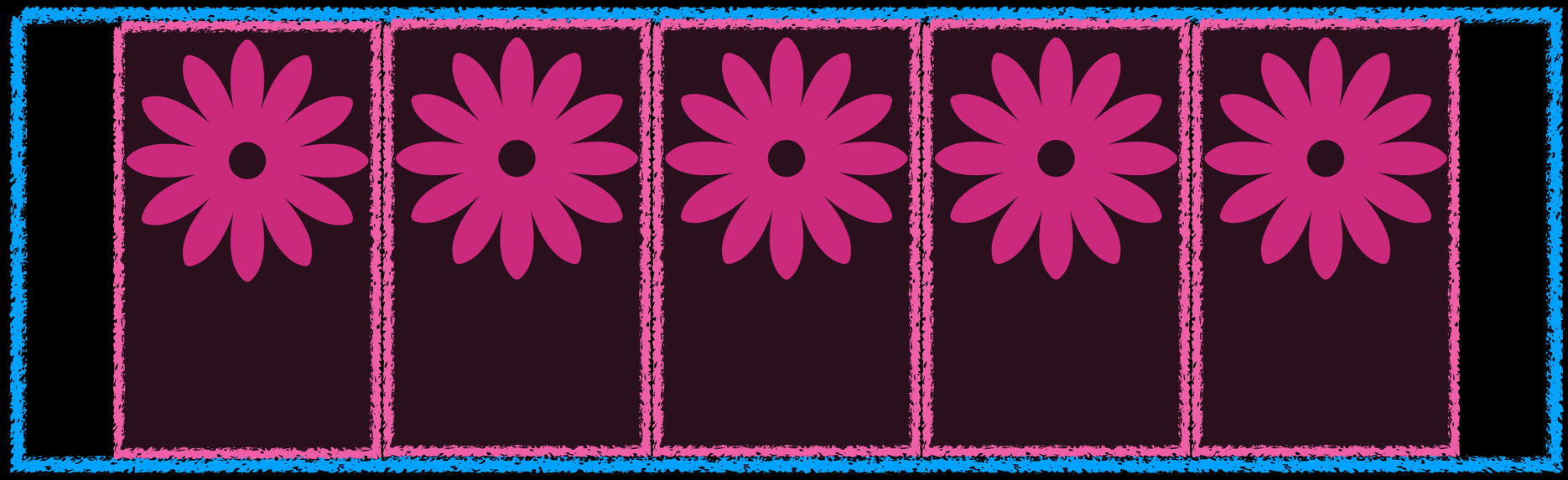
```
display: flex;
justify-content: center;
height: 400px;
```

Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용



```
display: flex;
justify-content: center;
height: 400px;
```

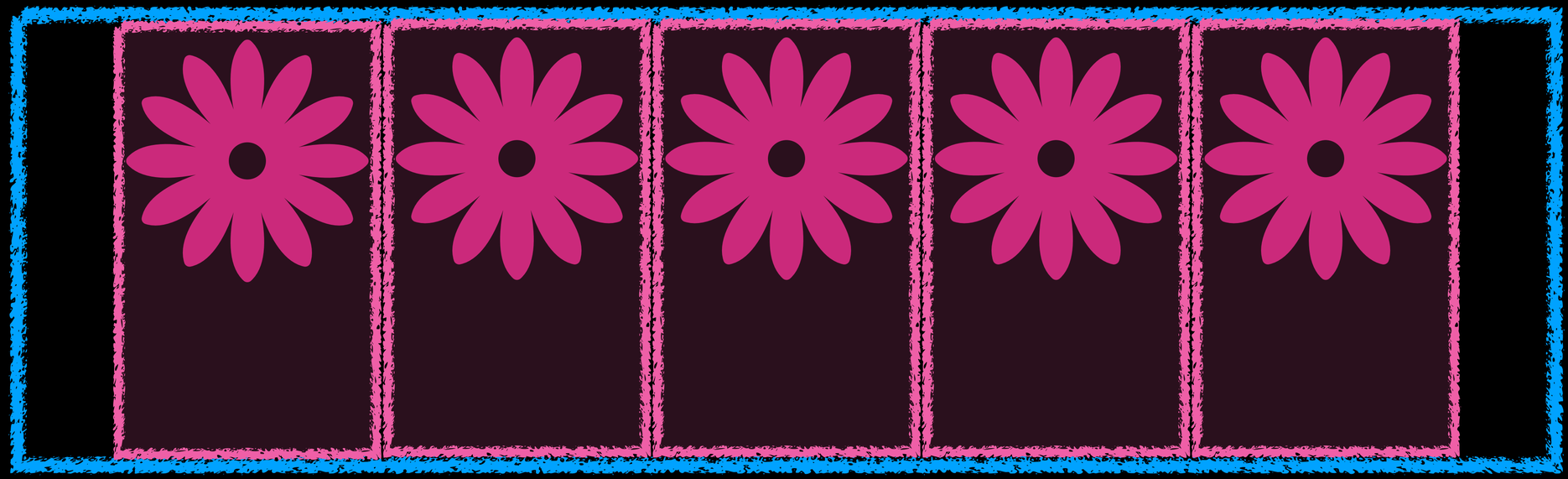
Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용

Cross Axis



display: flex;
justify-content: center;
height: 400px;
align-items: normal stretch로 계산

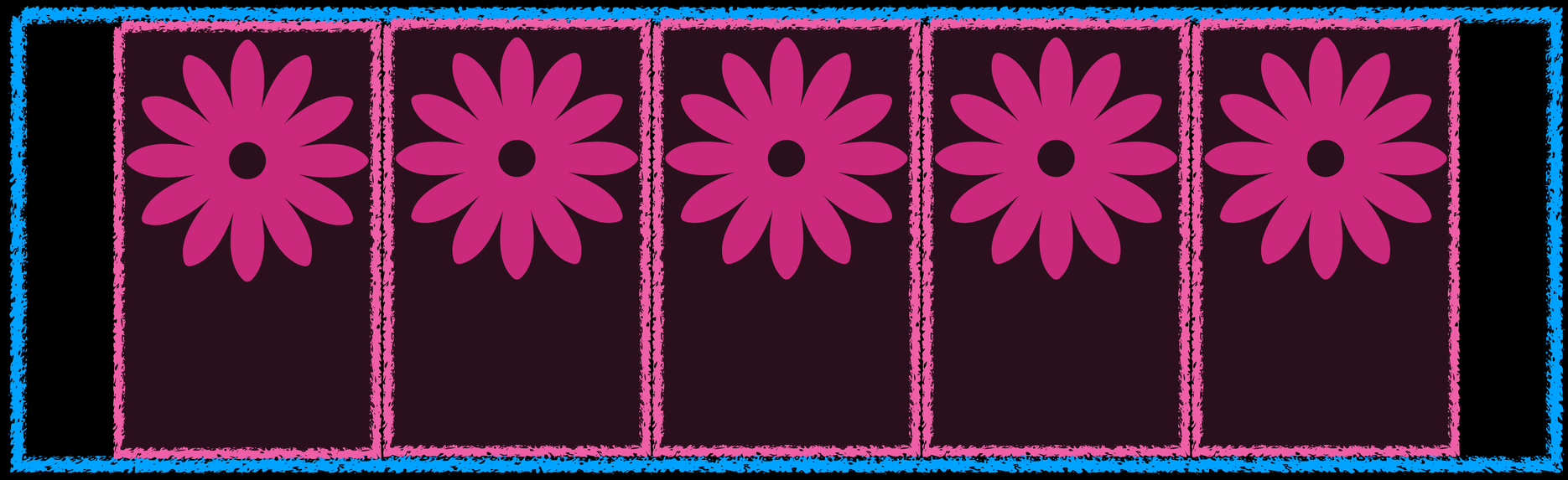
Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용

Cross Axis



```
display: flex;
justify-content: center;
height: 400px;
align-items: normal
```

Cross Axis 기준에서의 정렬 방법: align-items 속성을 활용

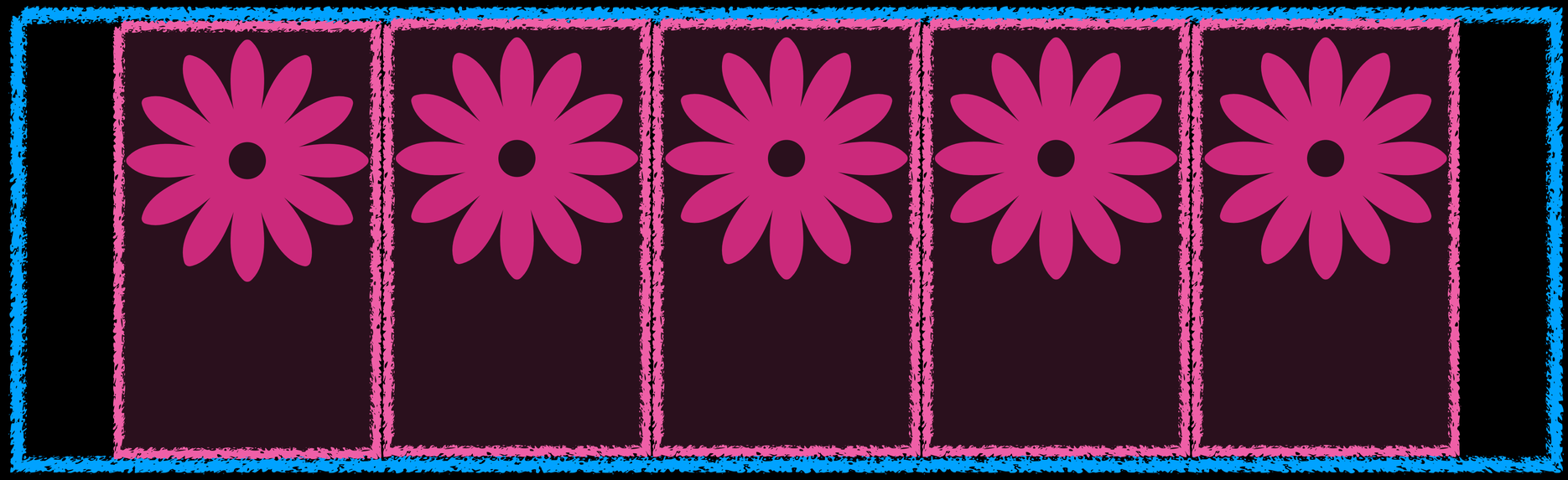
Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용

Cross Axis



```
display: flex;
justify-content: center;
height: 400px;
align-items: center
```

Cross Axis 기준에서의 정렬 방법: align-items 속성을 활용

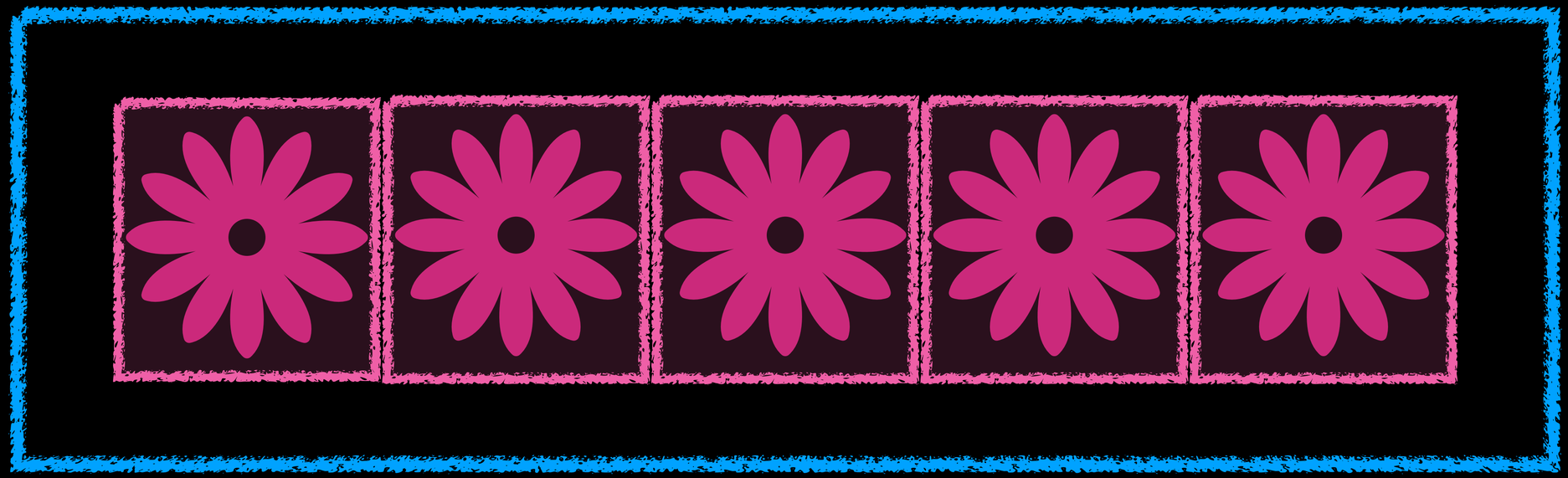
Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용

Cross Axis



```
display: flex;
justify-content: center;
height: 400px;
align-items: center
```

Cross Axis 기준에서의 정렬 방법: align-items 속성을 활용

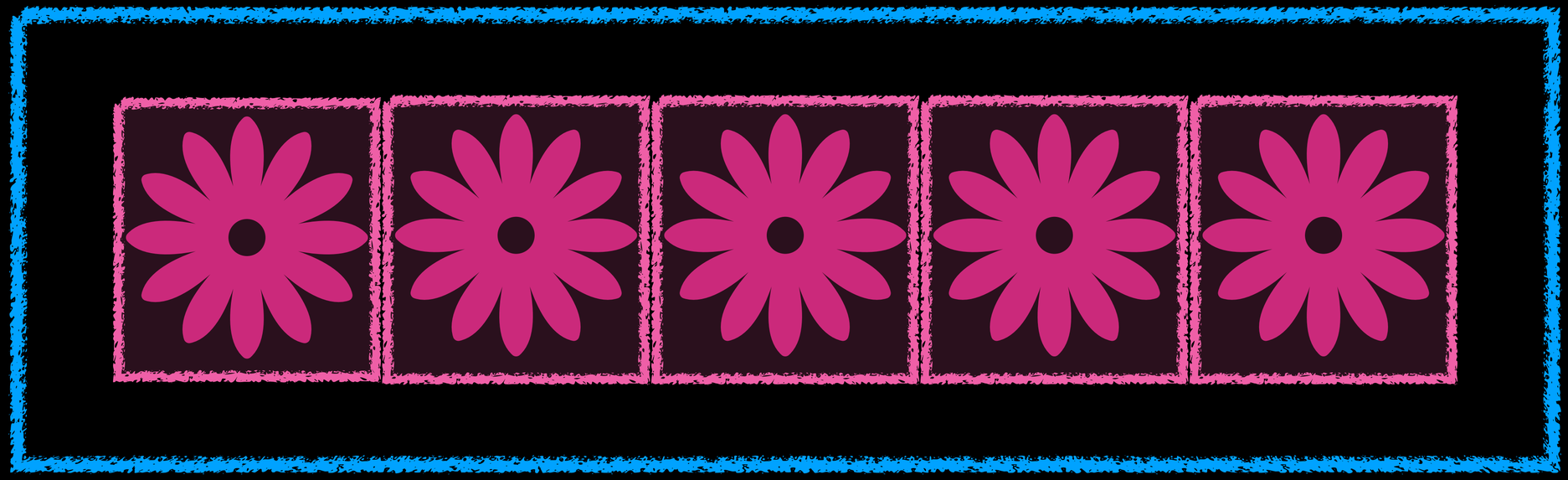
Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용

Cross Axis



```
display: flex;
justify-content: center;
height: 400px;
align-items: center
```

드디어 고정 height 높이 기준에서 내부 요소가 세로 중앙 정렬 가능하도록 할 수 있당

Cross Axis 기준에서의 정렬 방법: align-items 속성을 활용

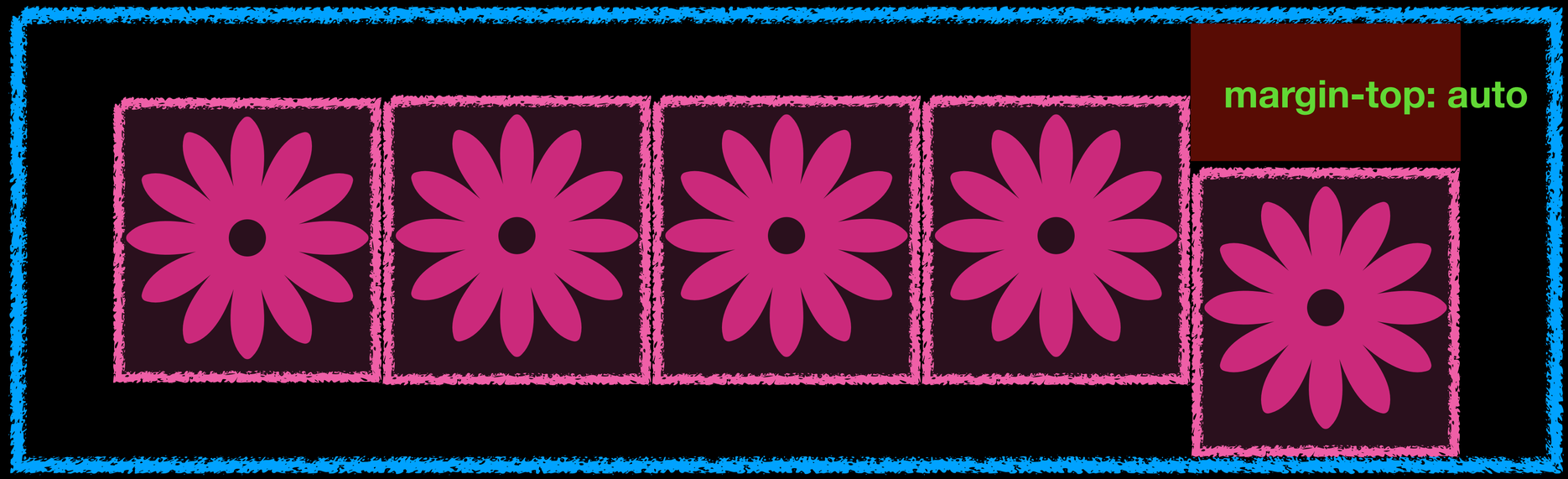
Flex Container

Flex Items

Main Axis

Main Axis 기준에서의 정렬 방법: justify-content 속성을 활용

Cross Axis




```
display: flex;
justify-content: center;
height: 400px;
align-items: center
```

Flex Container

- justify-content: Main Axis 기준에서의 정렬
- align-items: Cross Axis 기준에서의 정렬
- flex-wrap: 줄바꿈을 컨트롤
- gap: Flex Items 사이 간격, row-gap, column-gap의 단축 속성

- flex-direction: row

Flex Items

- flex-grow
- flex-shrink
- flex-basis
- min-width

Flex Items의
속성 3대장

- flex-grow
- flex-shrink
- flex-basis
- min-width

기본적으로 보여지는 크기, flex-grow, flex-shrink속성의 기준

flex-basis

flex-basis


```
.flex-item {
    flex-basis: auto;
}
```

초기값: 내부 콘텐츠 크기만큼으로 자동 조절

flex-basis


```
.flex-item {
    flex-basis: auto;
}
```

초기값: 내부 콘텐츠 크기만큼으로 자동 조절

flex-basis


```
.flex-item {
  flex-basis: auto;
}
```

초기값: 내부 콘텐츠 크기만큼으로 자동 조절

flex-basis


```
.flex-item {
  flex-basis: 0;
}
```

flex-basis


```
.flex-item {
  flex-basis: 0;
}
```

변화 없음

flex-basis


```
.flex-item {
  flex-basis: 0;
}
```

변화 없음

flex-basis


```
.flex-item {
  flex-basis: 0;
}
```

변화 없음

flex-basis


```
.flex-item {
  flex-basis: 0;
  min-width: auto;
}
```

변화 없음

이 녀석이 막고 있음

flex-basis


```
.flex-item {
  flex-basis: 0;
  min-width: auto;
}
```

변화 없음

Flex Item의 min-width: auto는 내부 콘텐츠 크기 만큼으로 계산된다.

flex-basis


```
.flex-item {
    flex-basis: 0;
    min-width: auto;
}
```

변화 없음

Flex Item의 min-width: auto는 내부 콘텐츠 크기 만큼으로 계산된다.
즉, 해당 요소는 자신의 콘텐츠보다 더 작아 질 수 없다.

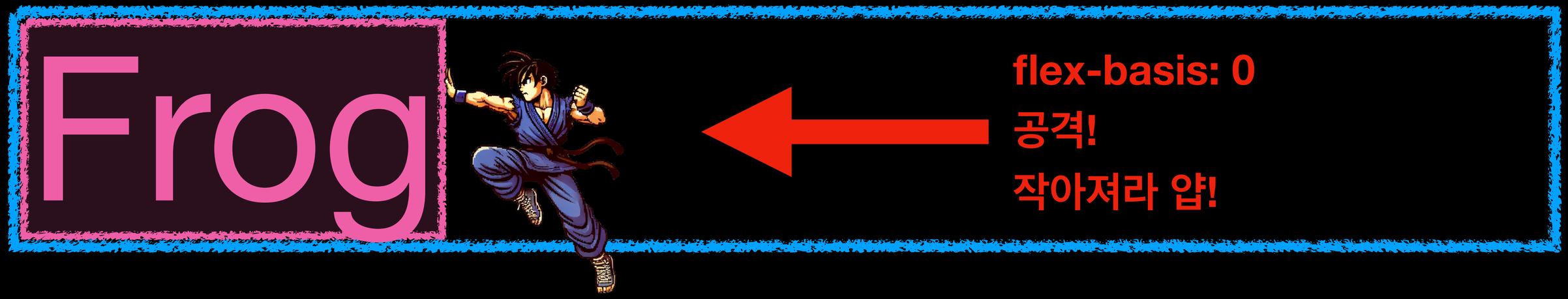
flex-basis


```
.flex-item {
    flex-basis: 0;
    min-width: auto;
}
```

변화 없음

Flex Item의 min-width: auto는 내부 콘텐츠 크기 만큼으로 계산된다.
즉, 해당 요소는 자신의 콘텐츠보다 더 작아 질 수 없다.

flex-basis

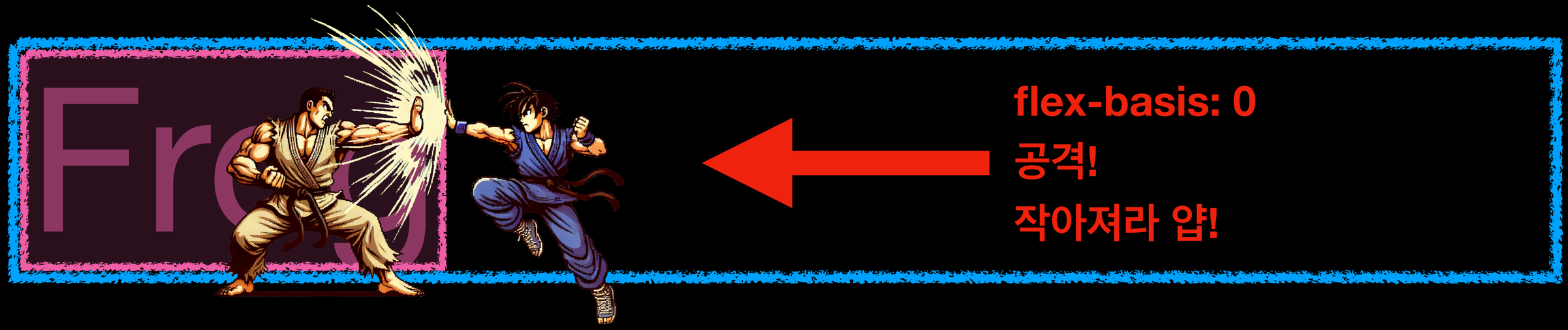



```
.flex-item {
  flex-basis: 0;
  min-width: auto;
}
```

작아지게 공격 드가자~

Flex Item의 min-width: auto는 내부 콘텐츠 크기 만큼으로 계산된다.
즉, 해당 요소는 자신의 콘텐츠보다 더 작아 질 수 없다.

flex-basis

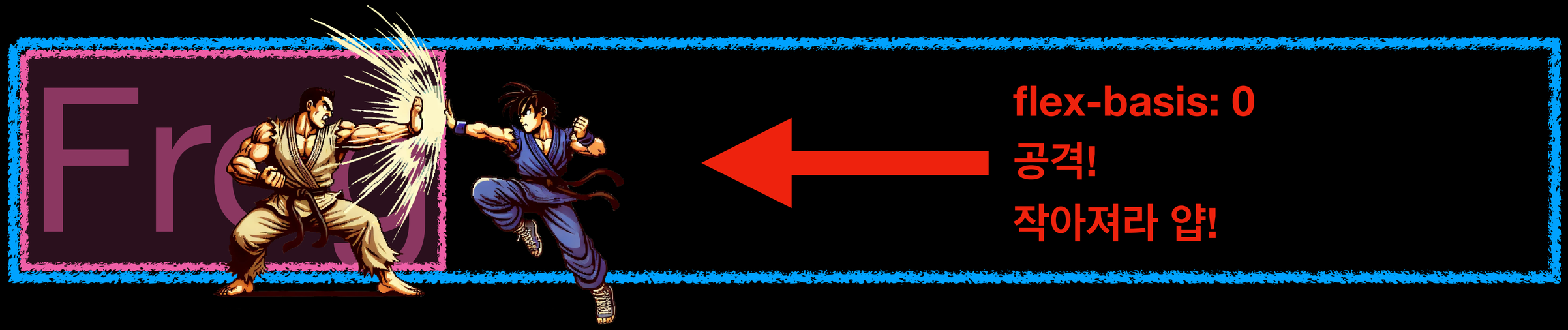



```
.flex-item {
    flex-basis: 0;
    min-width: auto;
}
```

min-width: auto에 의해 공격이 막힘!

Flex Item의 min-width: auto는 내부 콘텐츠 크기 만큼으로 계산된다.
즉, 해당 요소는 자신의 콘텐츠보다 더 작아 질 수 없다.

flex-basis

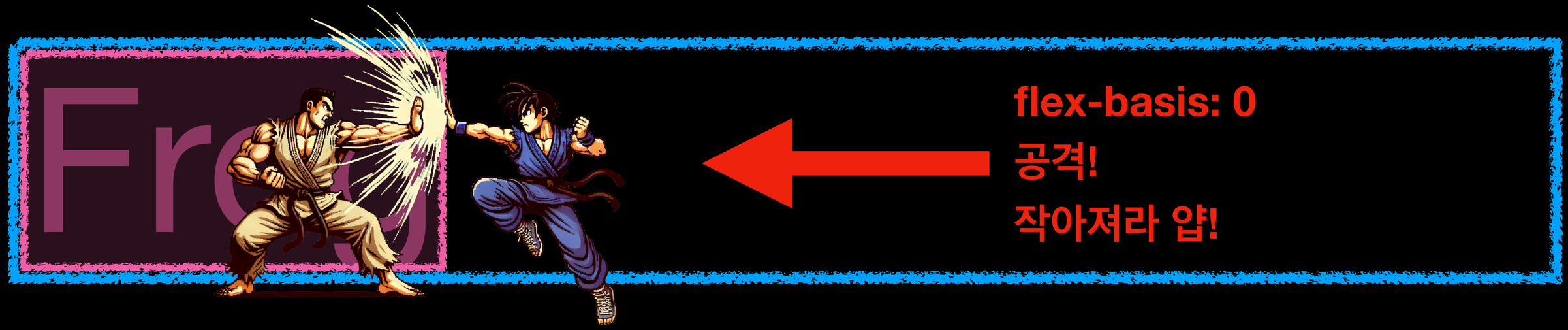



```
.flex-item {
  flex-basis: 0;
  min-width: 0;
}
```

min-width: auto에 의해 공격이 막힘!

min-width: auto를 없앤다면?

flex-basis




```
.flex-item {
  flex-basis: 0;
  min-width: 0;
}
```

min-width: auto에 의해 공격이 막힘!

min-width: auto를 없앤다면?

작아질 수 있음

flex-basis


```
.flex-item {
  flex-basis: 0;
  min-width: 0;
}
```

min-width: auto에 의해 공격이 막힘!

min-width: auto를 없앤다면?

작아질 수 있음

flex-basis


```
.flex-item {
  flex-basis: auto;
}
```

가용 공간을 비율로 나누어 flex-basis값에 확장!

flex-grow

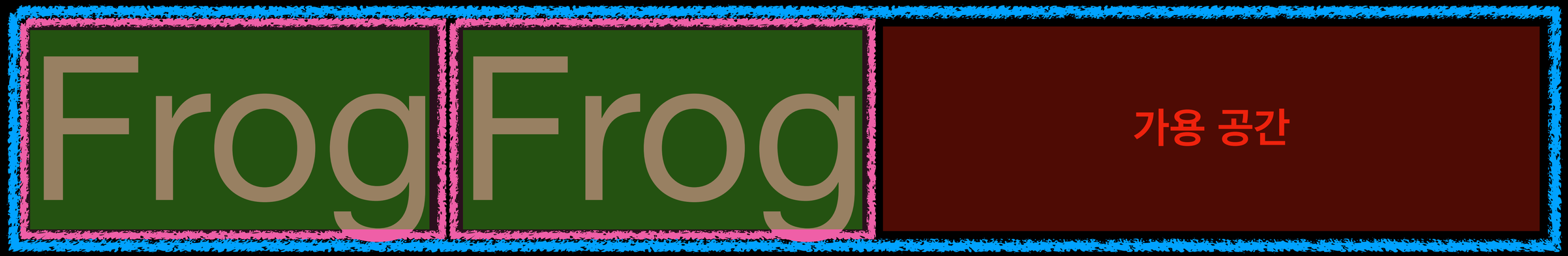
flex-grow


```
.flex-item {
  flex-basis: auto;
}
```

flex-grow

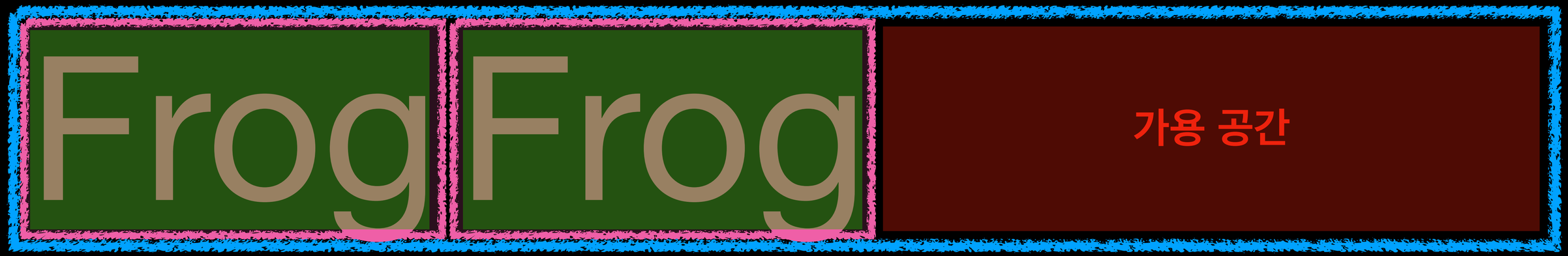

```
.flex-item {
  flex-basis: auto;
}
```

flex-grow




```
.flex-item {
  flex-basis: auto;
}
```

flex-grow

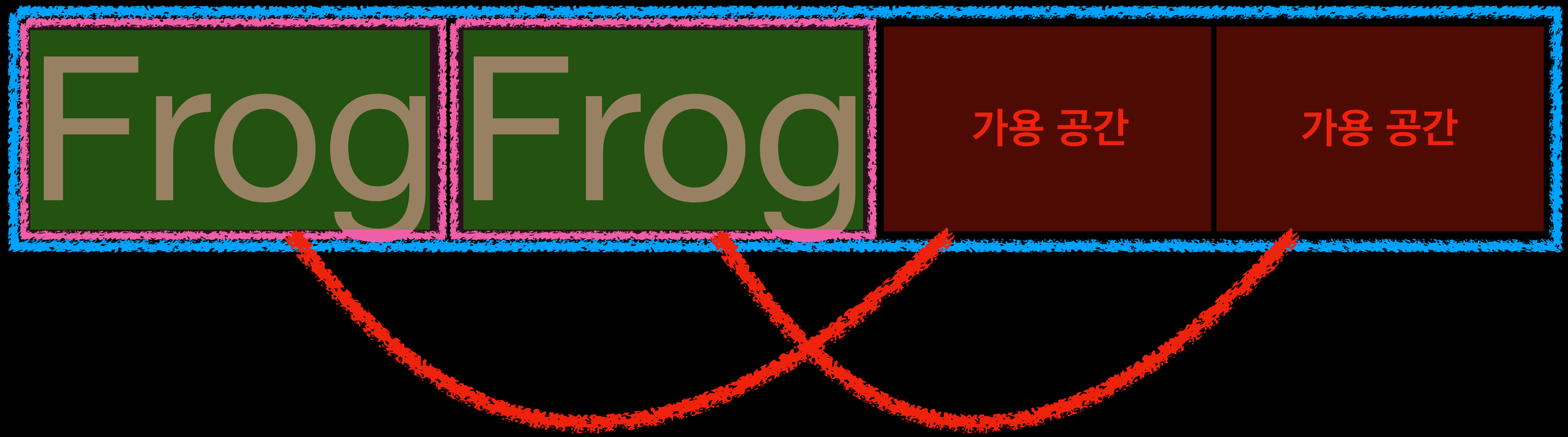



```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow

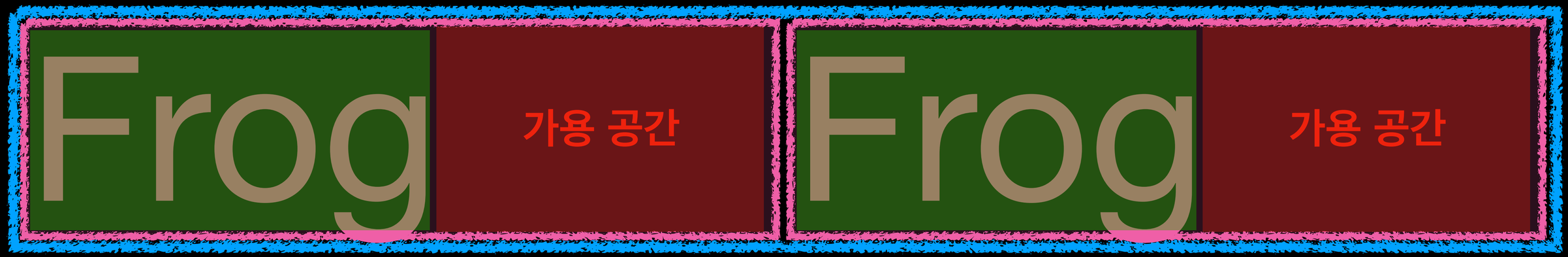

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



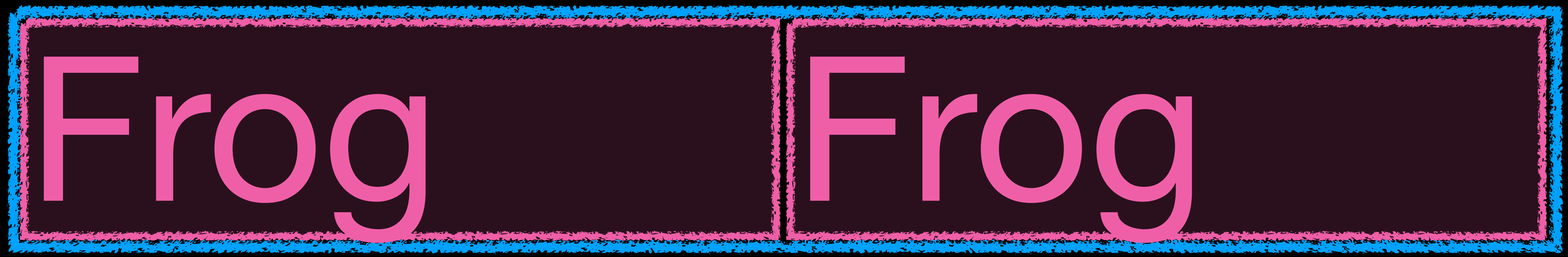

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



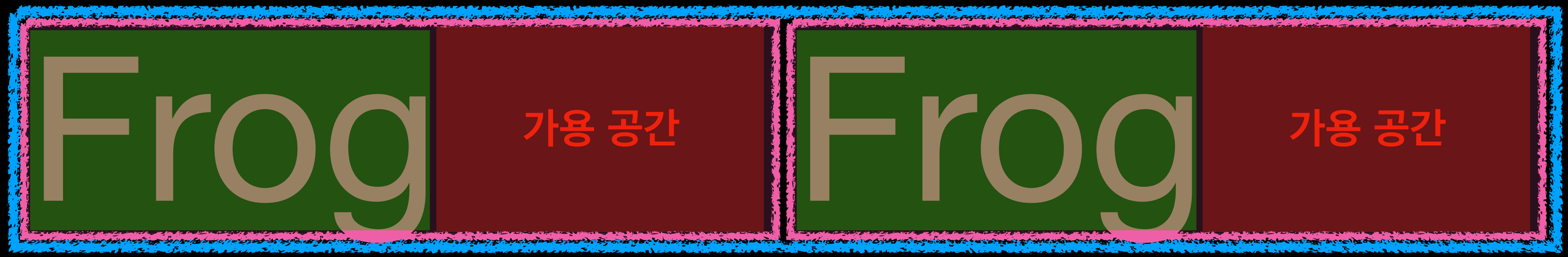

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



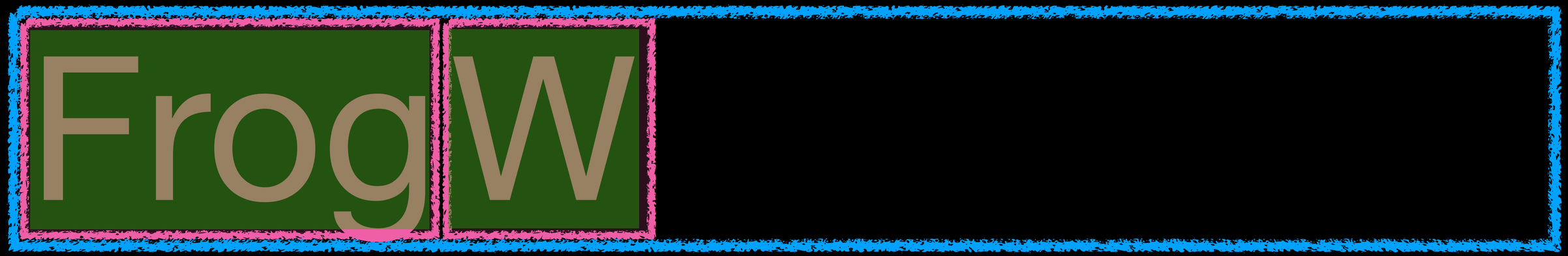

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



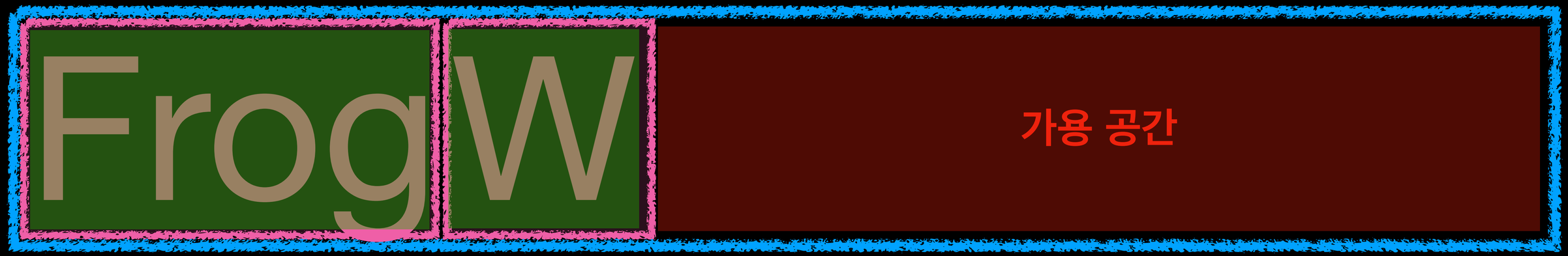

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



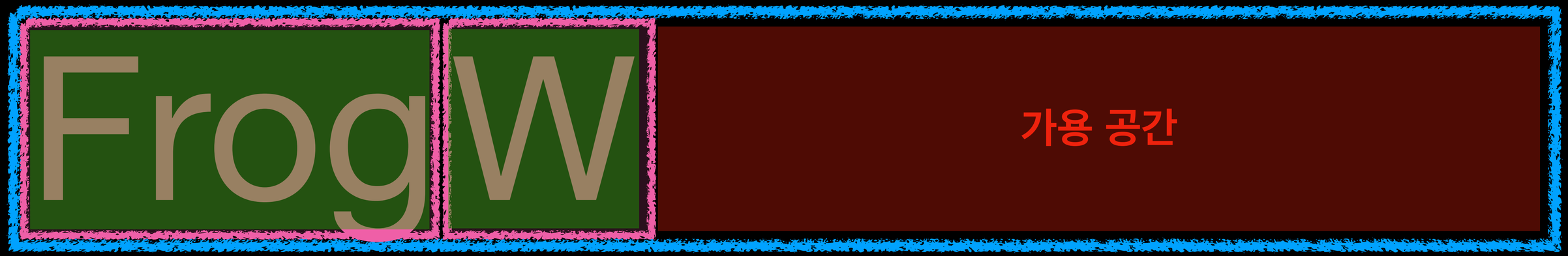

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow




```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow

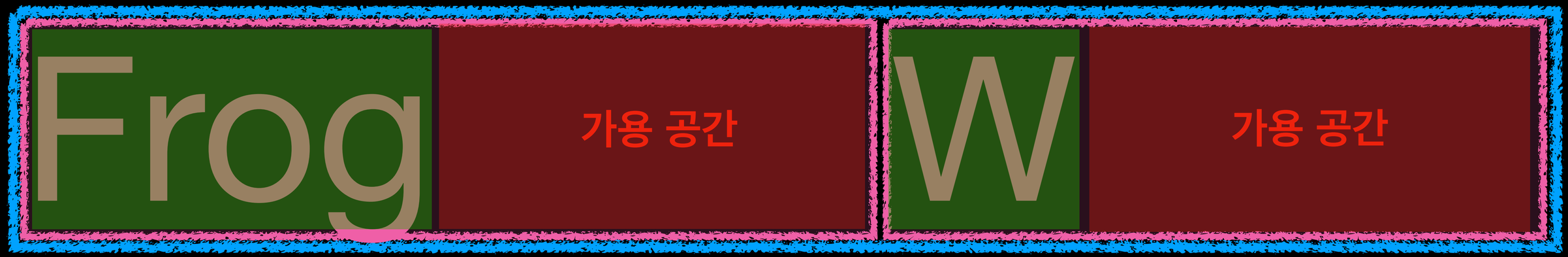



```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow

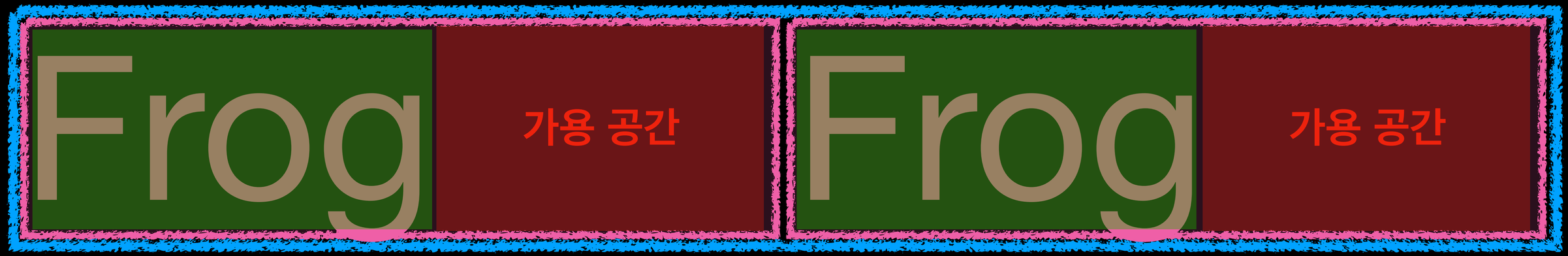

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



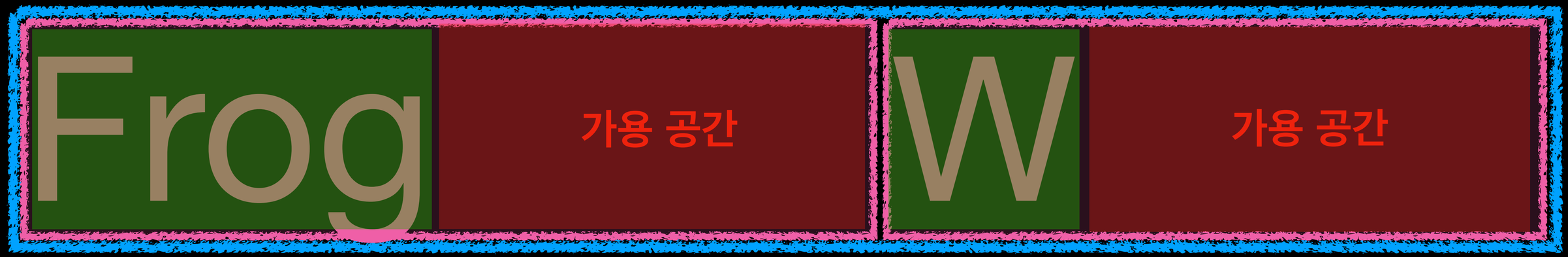

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



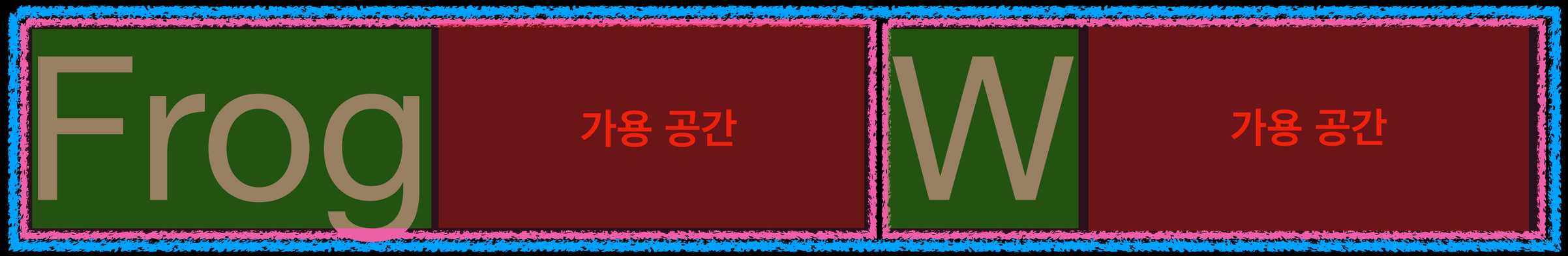

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



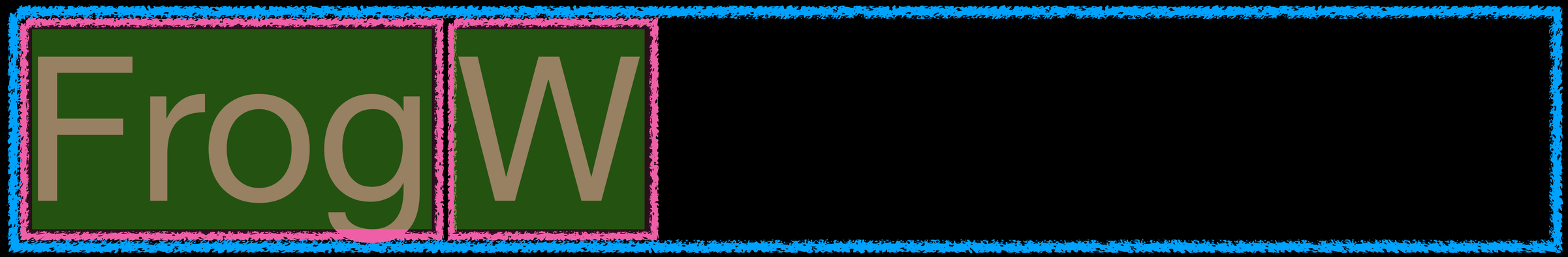

```
.flex-item {
  flex-basis: auto;
  flex-grow: 1;
}
```

flex-grow



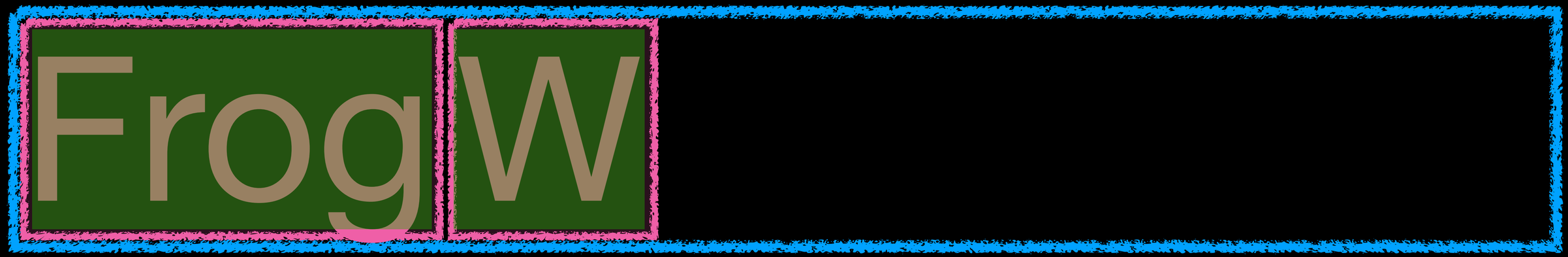

```
.flex-item {
  flex-basis: auto;
  flex-grow: 0;
}
```

flex-grow



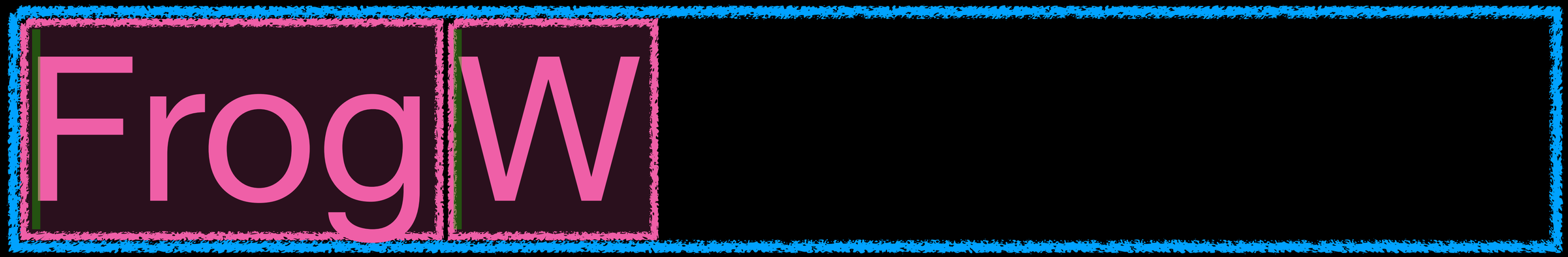

```
.flex-item {
  flex-basis: auto;
  flex-grow: 0;
}
```

flex-grow



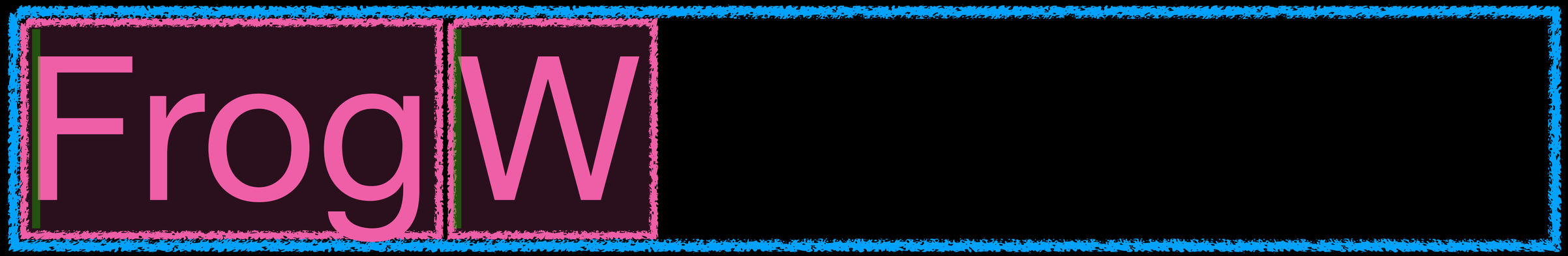

```
.flex-item {
  flex-basis: 0;
  flex-grow: 0;
}
```

flex-grow




```
.flex-item {
  flex-basis: 0;
  flex-grow: 0;
}
```

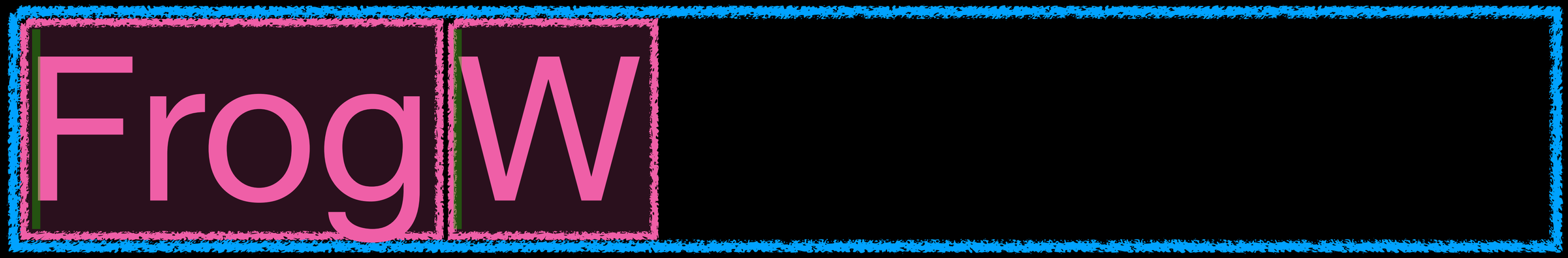
flex-grow



내부적으로 flex-basis: 0은 적용이 된 상태이지만 min-width: auto에 의해 시각적인 변화는 없음

```
.flex-item {
  flex-basis: 0;
  flex-grow: 0;
}
```

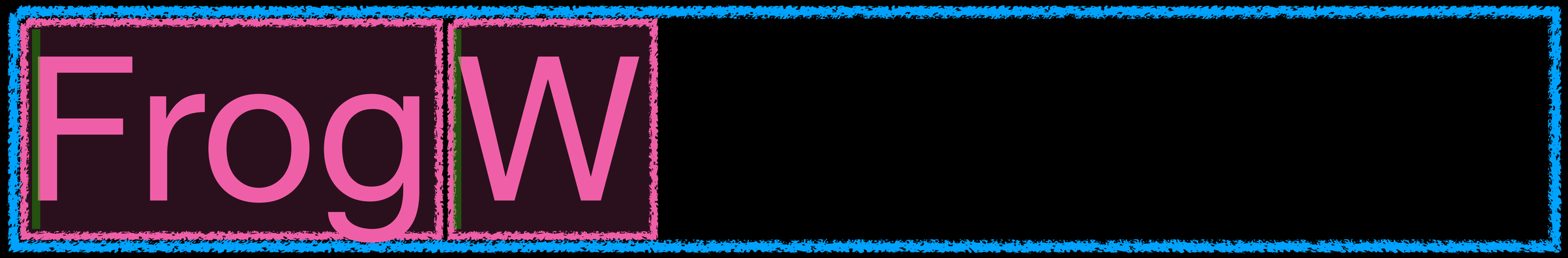
flex-grow



내부적으로 flex-basis: 0은 적용이 된 상태이지만 min-width: auto에 의해 시각적인 변화는 없음

```
.flex-item {
  flex-basis: 0;
  flex-grow: 1;
}
```

flex-grow

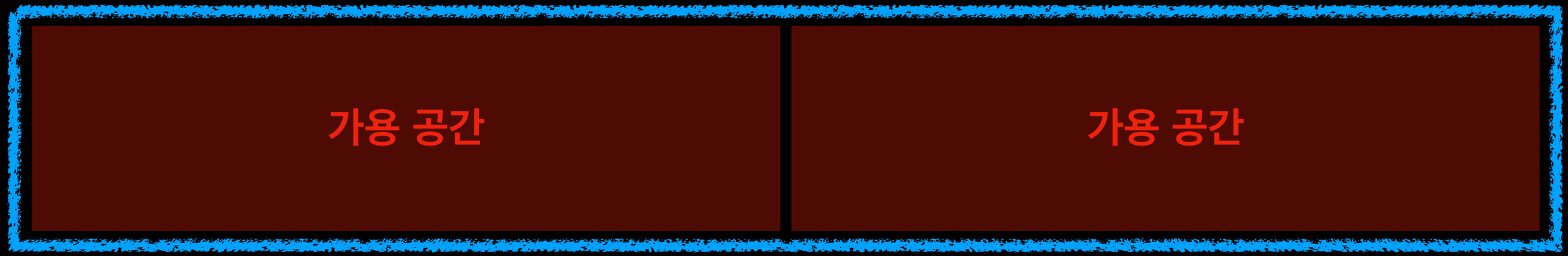


내부적으로 flex-basis: 0은 적용이 된 상태이지만 min-width: auto에 의해 시각적인 변화는 없음

```
.flex-item {
  flex-basis: 0;
  flex-grow: 1;
}
```

이제 flex-grow: 1로만 Flex Item의 너비가 결정!

flex-grow

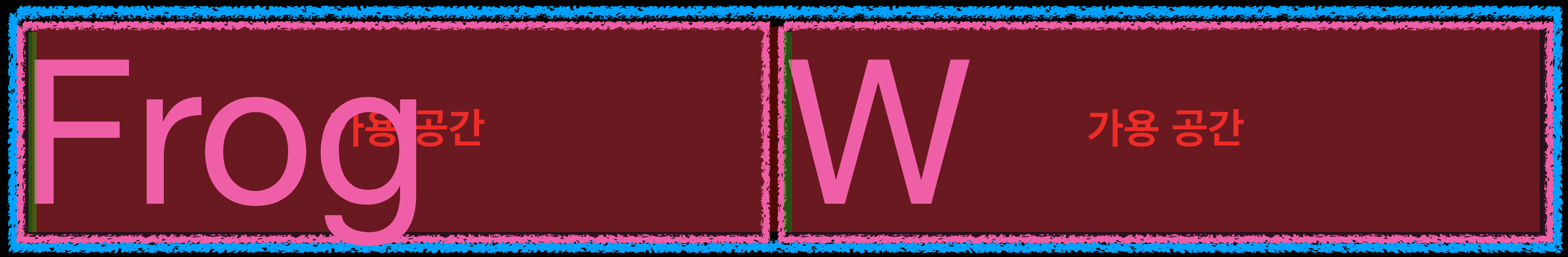


내부적으로 flex-basis: 0은 적용이 된 상태이지만 min-width: auto에 의해 시각적인 변화는 없음

```
.flex-item {
  flex-basis: 0;
  flex-grow: 1;
}
```

이제 flex-grow: 1로만 Flex Item의 너비가 결정!

flex-grow

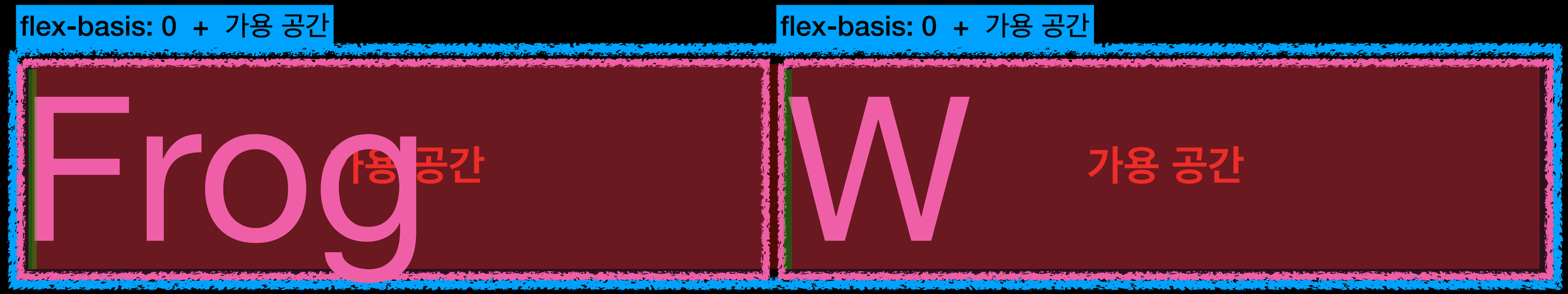


내부적으로 flex-basis: 0은 적용이 된 상태이지만 min-width: auto에 의해 시각적인 변화는 없음

```
.flex-item {
  flex-basis: 0;
  flex-grow: 1;
}
```

이제 flex-grow: 1로만 Flex Item의 너비가 결정!

flex-grow



내부적으로 flex-basis: 0은 적용이 된 상태이지만 min-width: auto에 의해 시각적인 변화는 없음

```
.flex-item {
  flex-basis: 0;
  flex-grow: 1;
}
```

이제 flex-grow: 1로만 Flex Item의 너비가 결정!

flex-grow

내부적으로 flex-basis: 0은 적용이 된 상태이지만 min-width: auto에 의해 시각적인 변화는 없음

```
.flex-item {
  flex-basis: 0;
  flex-grow: 1;
}
```

이제 flex-grow: 1로만 Flex Item의 너비가 결정!

flex-grow


```
.flex-item {
  flex-basis: 0;
  flex-grow: 1;
}
```

내부 콘텐츠와 관계 없어짐!

이제 flex-grow: 1로만 Flex Item의 너비가 결정!

유연한 레이아웃을 위해 지정된 flex-basis값을 수축시키는 경우가 있다!
각 Flex Items가 수축 되는 비율을 설정

flex-shrink

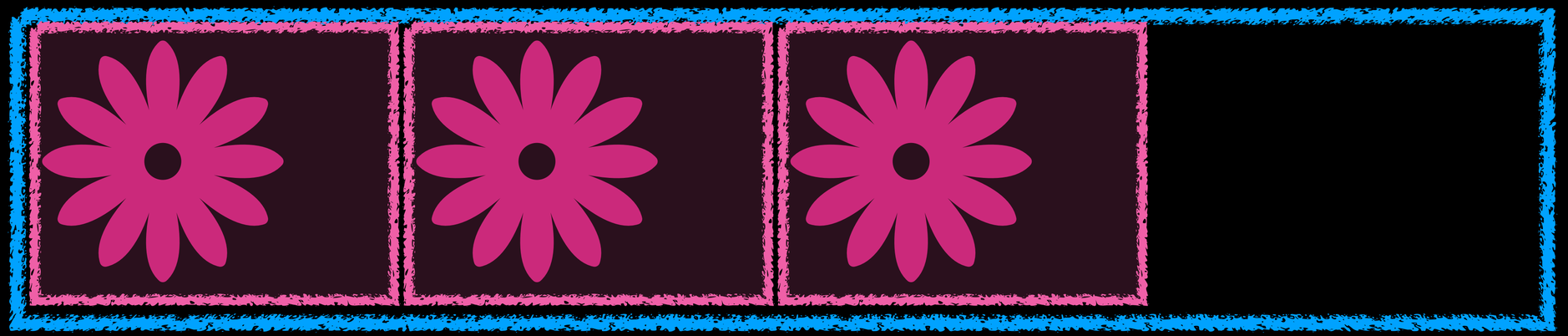
flex-shrink

```
.flex-item {
  flex-basis: 200px;
  flex-shrink: 1; 초기값: 1
}
```

flex-shrink

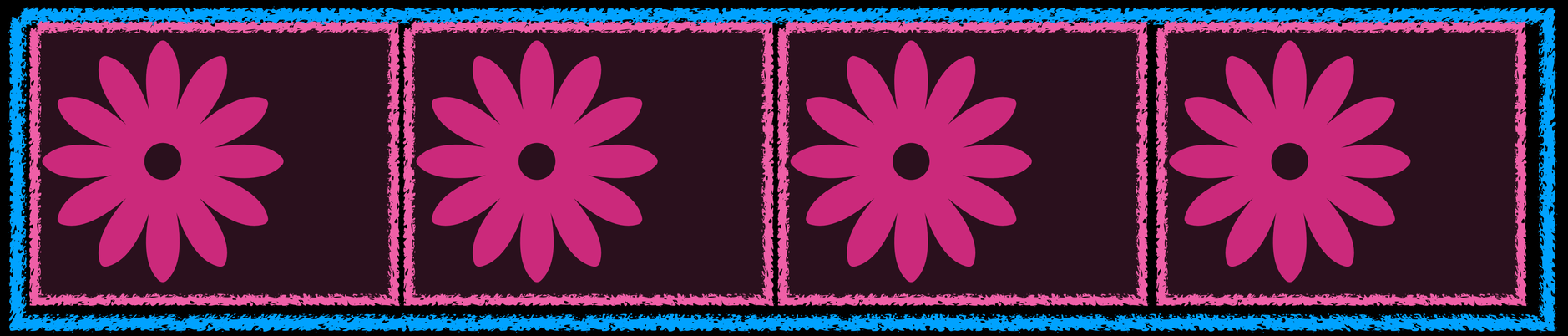
```
.flex-item {
  flex-basis: 200px;
  flex-shrink: 1; 초기값: 1
}
```

flex-shrink



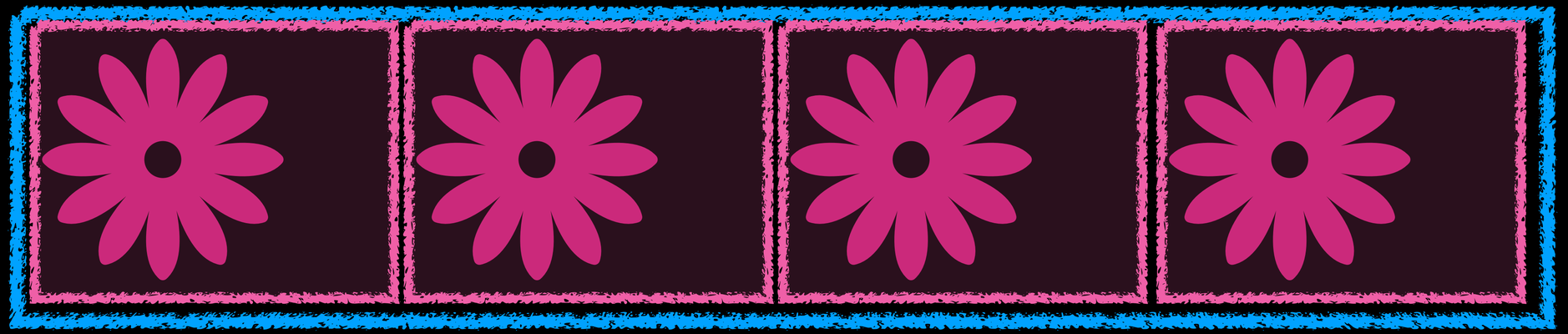
```
.flex-item {
  flex-basis: 200px;
  flex-shrink: 1; 초기값: 1
}
```

flex-shrink



```
.flex-item {
  flex-basis: 200px;
  flex-shrink: 1; 초기값: 1
}
```

flex-shrink

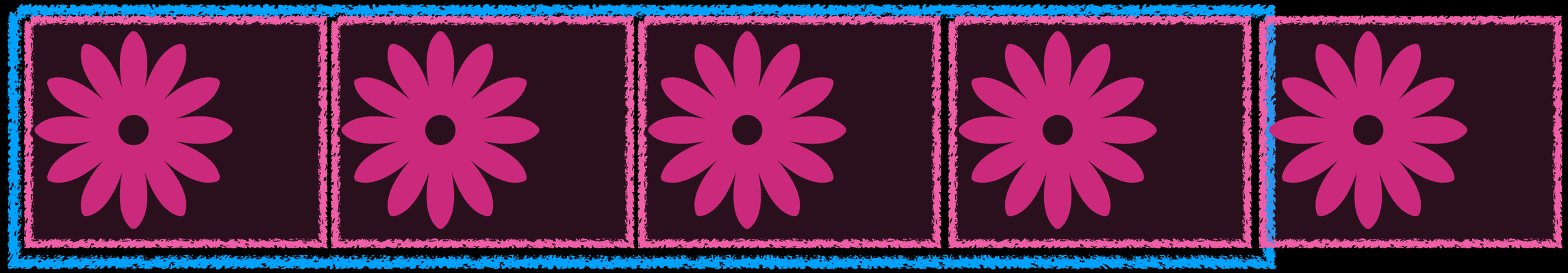


```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화로운 시기에 flex-shrink: 1은 실업자다!

flex-shrink

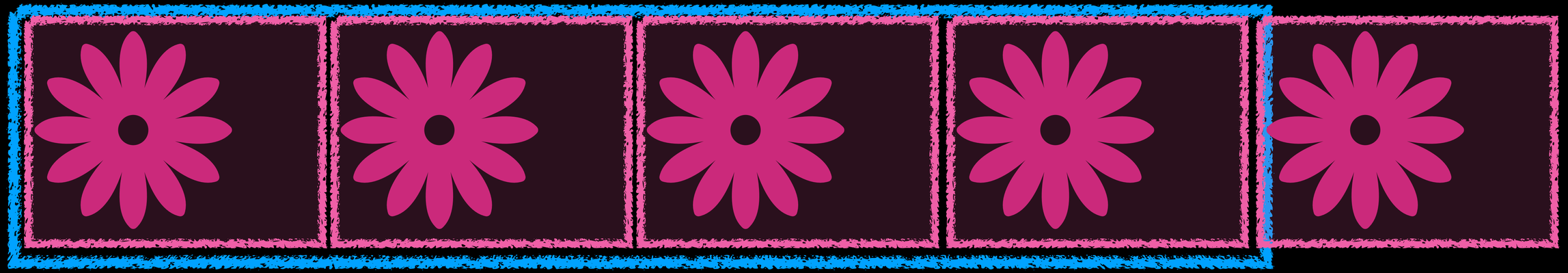


```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화로운 시기에 flex-shrink: 1은 실업자다!

flex-shrink

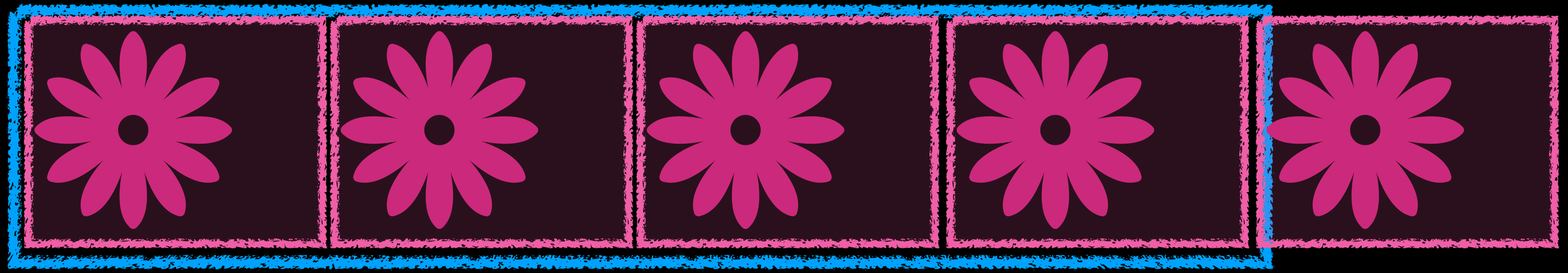


```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화를 깨버리자!
콘텐츠가 많아져서 공간이 부족한 상황!

flex-shrink

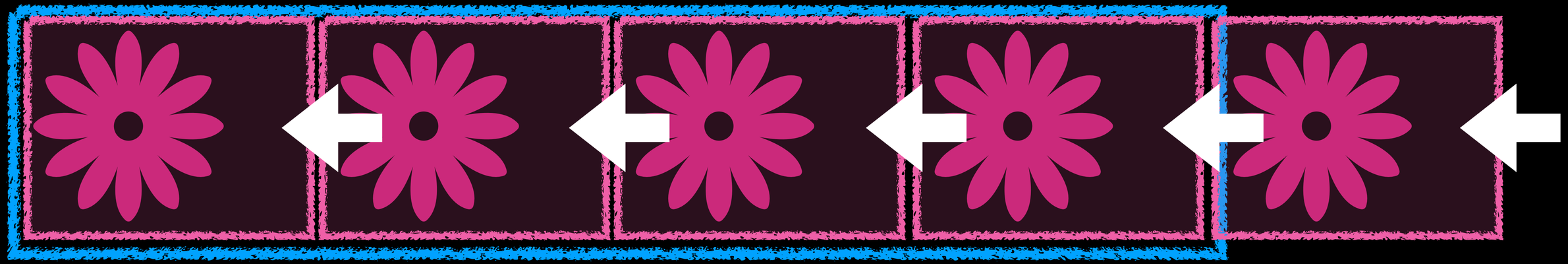


```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화를 유지하기 위해
Flexbox는 각 Flex Items의 flex-basis값을
같은 비율로 수축 시킨다!

flex-shrink

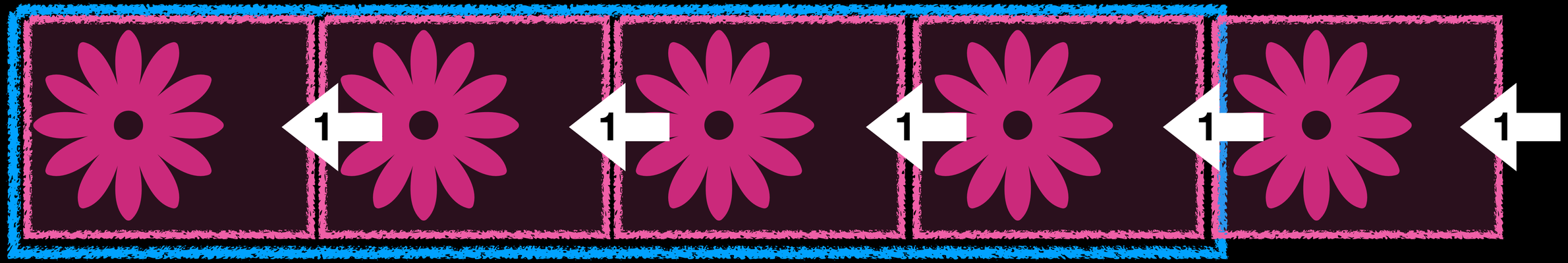


```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화를 유지하기 위해
Flexbox는 각 Flex Items의 flex-basis값을
같은 비율로 수축 시킨다!

flex-shrink

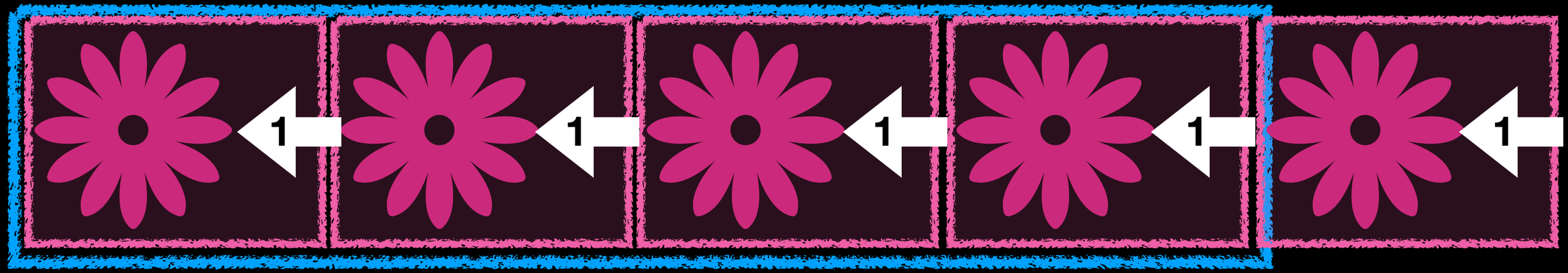



```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화를 유지하기 위해
Flexbox는 각 Flex Items의 flex-basis값을
같은 비율로 수축 시킨다!

flex-shrink

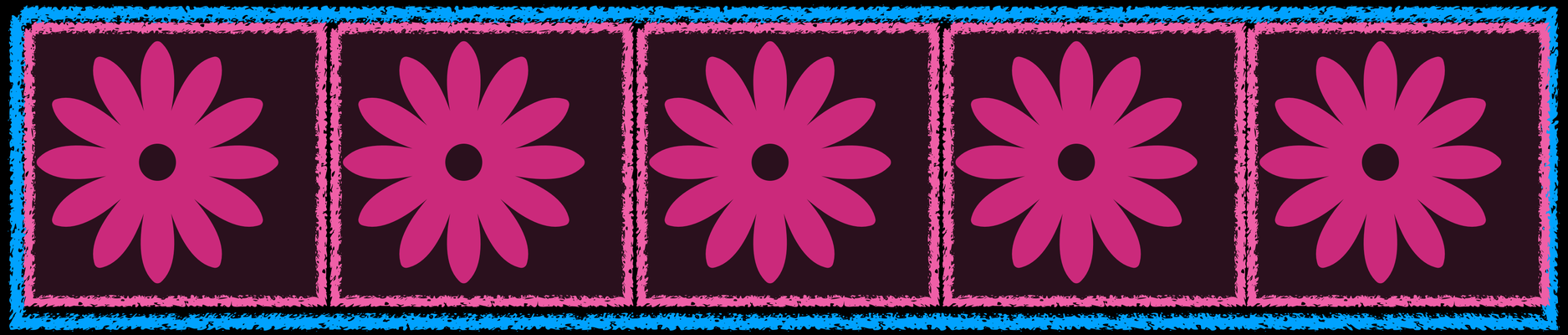



```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화를 유지하기 위해
Flexbox는 각 Flex Items의 flex-basis값을
같은 비율로 수축 시킨다!

flex-shrink

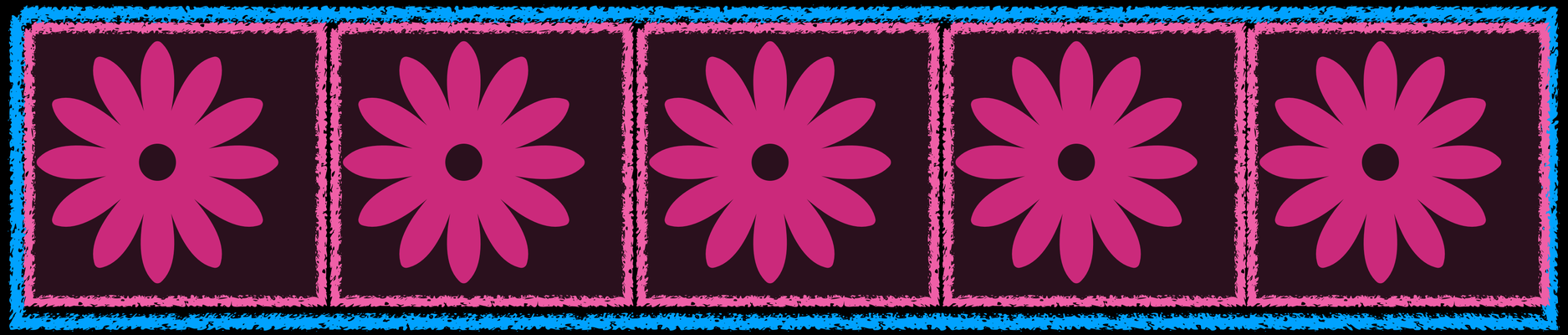


```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1

평화를 유지하기 위해
Flexbox는 각 Flex Items의 flex-basis값을
같은 비율로 수축 시킨다!

flex-shrink



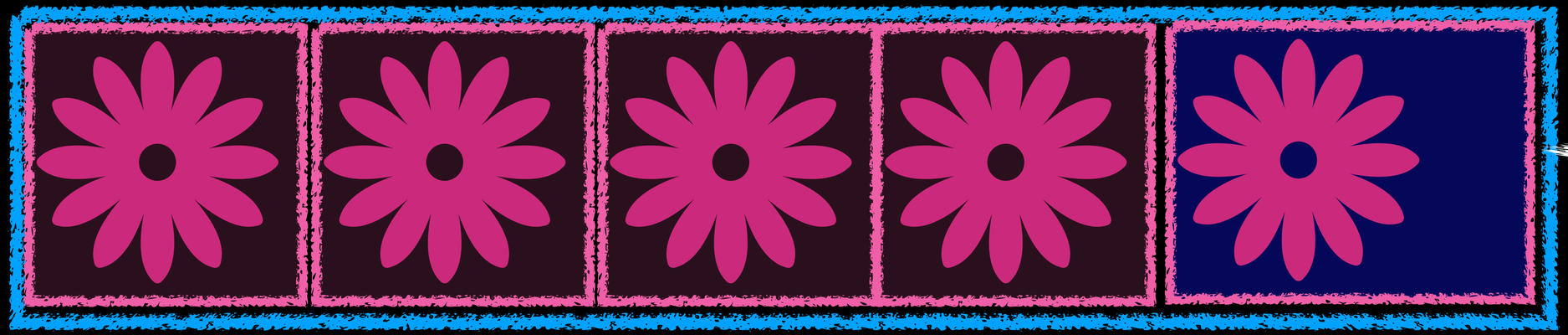
```
.flex-item {
  flex-basis: 200px;
  flex-shrink: 1;
}
```

초기값: 1

이것이 평화!

Flexible한 레이아웃!

flex-shrink



마지막꺼 하나만 수축되지 않게!
flex-shrink: 0;
flex-basis: 200px이 보장된다!

```
.flex-item {
    flex-basis: 200px;
    flex-shrink: 1;
}
```

초기값: 1