10. CSS Flexbox

Flex 컨테이너 vs 아이템

CSS Flexbox는 **1차원 레이아웃**을 만들기 위한 강력한 도구다.
`flexbox`를 사용하기 위해서는 먼저 **Flex 컨테이너와 Flex 아이템의 역할과 차이**를 명확히 이해해야 한다.

---

1. 기본 용어 정의

용어	설명
Flex 컨테이너	`display: flex` 또는 `display: inline-flex`가 설정된 요소
Flex 아이템	Flex 컨테이너의 **직계 자식 요소들**

---

2. Flex 컨테이너의 역할

Flex 컨테이너는 **자식 요소를 유연하게 배치**하는 환경을 제공한다.
이를 위해 컨테이너는 **축을 설정하고**, 자식의 위치, 정렬, 줄바꿈 등을 제어한다.

주요 속성 (Flex 컨테이너용)

속성	역할
`display: flex`	Flexbox 레이아웃 시작
`flex-direction`	주축 방향 설정 (row, column 등)
`flex-wrap`	줄바꿈 허용 여부
`justify-content`	주축 방향 정렬 방식
`align-items`	교차축 방향 정렬 방식
`align-content`	여러 줄의 교차축 정렬

---

3. Flex 아이템의 역할

Flex 아이템은 Flex 컨테이너 내부의 **직계 자식 요소들**로,
자신의 크기와 비율을 조절하며 주어진 공간 안에서 유연하게 배치된다.

주요 속성 (Flex 아이템용)

속성	역할
`flex-grow`	여유 공간을 나눠 가짐

속성	역할
`flex-shrink`	공간 부족 시 줄어드는 비율
`flex-basis`	초기 기본 크기 설정
`flex`	위 3개를 축약한 속성 (`flex: 1 1 0`)
`align-self`	개별 아이템의 교차축 정렬
`order`	아이템의 시각적 순서 지정

---

4. 예제

```
<div class="container">
  <div class="item">1</div>
  <div class="item">2</div>
  <div class="item">3</div>
</div>
```

```
.container {
  display: flex; /* Flex 컨테이너 */
  justify-content: space-between;
  align-items: center;
}

.item {
  flex: 1; /* Flex 아이템 */
  padding: 10px;
}
```

---

5. 차이 요약표

항목	Flex 컨테이너	Flex 아이템
정의	부모 요소	자식 요소
필수 속성	`display: flex`	없음 (자동으로 적용됨)
주요 역할	전체 레이아웃 제어	공간 배분, 크기 조정
위치	최상위 (자식의 부모)	컨테이너의 직계 자식
제어 대상	자식들의 배치 방향, 정렬	개별 아이템의 크기 및 정렬

---

6. 주의 사항

- Flex 아이템은 **Flex 컨테이너의 직계 자식만** 해당된다
- Flex 아이템 안에 또 다른 `display: flex`가 있다면 **중첩된 Flex 구조**도 가능하다
- Flex 컨테이너에 `min-height`, `max-width` 등을 함께 지정하면 반응형 설계가 더욱 용이하다

결론

> Flex 컨테이너는 **레이아웃을 지휘**하는 부모,
> Flex 아이템은 **배치와 비율 조절에 순응하는 자식**이다.
> 두 역할을 구분하면 Flexbox를 훨씬 명확하고 강력하게 활용할 수 있다.

주요 속성

Flexbox에서는 **컨테이너와 아이템 각각에 적용하는 속성**이 다르다.
이 섹션에서는 **레이아웃 설계에서 가장 자주 사용되는 핵심 속성들**을 중심으로,
컨테이너용과 아이템용을 구분하여 정리한다.

📌 1. Flex 컨테이너용 속성

컨테이너에 지정하면 **자식(Flex 아이템)들의 배치 방향, 정렬, 줄바꿈 등**을 제어함.

▸ `display`

```
.container {
  display: flex; /* 블록 레벨 Flex 컨테이너 */
  display: inline-flex; /* 인라인 레벨 Flex 컨테이너 */
}
```

▸ `flex-direction`

```
flex-direction: row;         /* 기본값, 수평 정렬 (왼→오) */
flex-direction: row-reverse; /* 수평 정렬 (오→왼) */
flex-direction: column;      /* 수직 정렬 (위→아래) */
flex-direction: column-reverse; /* 수직 정렬 (아래→위) */
```

→ **주축(main axis)** 설정

▸ `flex-wrap`

```
flex-wrap: nowrap;     /* 기본값, 한 줄에 배치 */
flex-wrap: wrap;       /* 여러 줄 허용 (넘치면 다음 줄로) */
flex-wrap: wrap-reverse; /* 줄바꿈 방향 반대로 */
```

▸ `justify-content`

> **주축 방향 정렬 (수평 or 수직 depending on `flex-direction`)**

```
justify-content: flex-start;    /* 시작 지점 정렬 (기본) */
justify-content: flex-end;      /* 끝 지점 정렬 */
justify-content: center;        /* 가운데 정렬 */
justify-content: space-between; /* 양 끝 정렬 + 사이 균등 */
justify-content: space-around;  /* 요소 간 간격 균등 */
justify-content: space-evenly;  /* 모든 간격 완전 균등 */
```

▸ `align-items`

> **교차축 방향 정렬 (수직 or 수평 depending on `flex-direction`)**

```
align-items: stretch;    /* 기본값, 컨테이너 높이에 맞게 늘어남 */
align-items: flex-start; /* 위쪽 정렬 */
align-items: flex-end;   /* 아래쪽 정렬 */
align-items: center;     /* 가운데 정렬 */
align-items: baseline;   /* 텍스트 기준선 맞춤 */
```

▸ `align-content`

> **여러 줄 정렬 시 교차축 기준 정렬** (`flex-wrap`이 있어야 동작)

```
align-content: flex-start;
align-content: flex-end;
align-content: center;
align-content: space-between;
align-content: space-around;
align-content: stretch;
```

📌 2. Flex 아이템용 속성

아이템 각각의 **성장, 축소, 정렬, 순서 등**을 제어함.

▸ `flex-grow`

> **여유 공간을 얼마나 차지할지** (비율)

```
.item {
  flex-grow: 1; /* 여유 공간을 동일 비율로 분배 */
}
```

▸ `flex-shrink`

> **공간 부족 시 얼마나 축소될 수 있는지** (비율)

```
.item {
  flex-shrink: 1; /* 기본값: 자동 축소 가능 */
}
```

▸ `flex-basis`

> **기본 크기 지정** (`width`와 유사하지만 Flexbox 컨텍스트에 따라 작동)

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

▸ `flex`

> 위 3개 속성의 단축형

```
flex: <grow> <shrink> <basis>;

.item {
  flex: 1 1 0;      /* grow=1, shrink=1, basis=0 */
  flex: 0 1 auto;   /* 기본값 */
  flex: 1;          /* grow=1, shrink=1, basis=0% */
}
```

▸ `align-self`

> **아이템 개별 교차축 정렬 지정**

```
.item {
  align-self: flex-start;
}
```

→ `align-items`의 개별 override

▸ `order`

> **아이템 시각적 순서 지정**

```
.item {
  order: 2; /* 숫자가 작을수록 먼저 나옴 (기본값 0) */
}
```

📌 3. 핵심 요약표

구분	속성	기능
컨테이너	`display`	Flexbox 모드 시작
컨테이너	`flex-direction`	주축 방향 설정
컨테이너	`flex-wrap`	줄바꿈 여부 설정
컨테이너	`justify-content`	주축 정렬
컨테이너	`align-items`	교차축 정렬
컨테이너	`align-content`	여러 줄 교차축 정렬
아이템	`flex-grow`	여유 공간 분배 비율
아이템	`flex-shrink`	공간 부족 시 축소 비율
아이템	`flex-basis`	기본 크기 설정
아이템	`flex`	위 3개 속성의 축약형
아이템	`align-self`	개별 교차축 정렬
아이템	`order`	시각적 순서 조정

결론

> Flexbox의 주요 속성들은 각 요소가 **유동적으로 크기와 정렬을 조절할 수 있게 해주는 도구**다.
> 컨테이너는 전체 흐름을, 아이템은 자신의 비율과 정렬을 담당하며,
> 이 두 축을 정복하면 거의 모든 레이아웃을 직관적이고 반응형으로 설계할 수 있다.

아이템 개별 속성

Flex 아이템의 개별 속성 정리

Flexbox에서 각 **아이템 단위로 조절할 수 있는 주요 속성들**은
레이아웃을 더 세밀하고 유연하게 설계할 수 있게 해준다.
이 속성들은 Flex 컨테이너 내부에서 **각 아이템이 공간을 어떻게 차지하고 정렬될지**를 제어한다.

---

1. `flex-grow`: 여유 공간의 성장 비율

- 컨테이너 안에 남는 공간이 있을 때, **얼마나 많이 늘어날 것인지** 지정
- 숫자가 클수록 더 많은 공간 차지
- 기본값: `0` (남는 공간을 차지하지 않음)

```css
.item {
  flex-grow: 1; /* 남는 공간을 다른 아이템과 1:1 비율로 나눔 */
}
```

- 예: `.item1 { flex-grow: 2; }`, `.item2 { flex-grow: 1; }`이면
  → `item1`이 `item2`보다 2배 넓음

---

2. `flex-shrink`: 공간 부족 시 축소 비율

- 컨테이너보다 아이템의 총 너비가 **커질 경우**, 얼마나 축소할 수 있는지를 지정
- 숫자가 클수록 더 많이 줄어듦
- 기본값: `1` (축소 가능)

```css
.item {
  flex-shrink: 1; /* 기본값, 공간 부족 시 줄어듦 */
}
```

- `flex-shrink: 0` → 줄어들지 않음 (overflow 가능)

---

3. `flex-basis`: 기본 크기 지정

- Flex 아이템의 **초기 크기(기준 크기)** 를 설정
- `width`와 비슷하지만, Flexbox에서 더 우선 적용됨
- 기본값: `auto` (내용 크기 또는 width 기준)

```css
.item {
  flex-basis: 200px; /* 최소 200px 확보 후 grow/shrink */
}
```

---

4. `flex`: 단축 속성 (`grow shrink basis`)

- 위의 3가지를 한 줄로 설정
- 자주 쓰이는 값:

```css
.item {
  flex: 1;        /* grow:1, shrink:1, basis:0% */
  flex: 0 1 auto; /* 기본값 */
}
```

---

5. `align-self`: 개별 아이템 정렬

- 컨테이너의 `align-items`를 무시하고,
  특정 아이템만 교차축에서 **개별 정렬**할 수 있게 함

```
.item {
  align-self: center;
}
```

- 가능한 값: `auto`, `flex-start`, `flex-end`, `center`, `baseline`, `stretch`

---

6. `order`: 아이템의 시각적 순서 설정

- HTML 순서와 무관하게 **레이아웃 순서를 변경**할 수 있음
- 숫자가 작을수록 앞에 나옴
- 기본값: `0`

```
.item1 {
  order: 2;
}
.item2 {
  order: 1;
}
```

→ `item2`가 시각적으로 먼저 배치됨

예제 요약

```
.item {
  flex-grow: 1;
  flex-shrink: 0;
  flex-basis: 100px;
  align-self: flex-end;
  order: 2;
}
```

요약표

속성	설명	기본값
`flex-grow`	남는 공간에 대해 성장 비율	`0`
`flex-shrink`	공간 부족 시 줄어드는 비율	`1`
`flex-basis`	초기 기본 크기	`auto`

속성	설명	기본값
`flex`	위 3개 속성의 축약형	`0 1 auto`
`align-self`	개별 아이템 정렬	`auto`
`order`	시각적 순서 조정	`0`

결론

> Flex 아이템 개별 속성은 **개성 있는 배치**를 만들고
> 동일한 컨테이너 안에서도 **다양한 크기, 정렬, 순서**를 지정할 수 있게 해준다.
> 특히 `flex`, `order`, `align-self`는 **복잡한 UI 구성에 유용한 도구**다.

축 개념 (Main axis / Cross axis)

Flexbox의 축 개념: Main Axis와 Cross Axis

Flexbox 레이아웃을 정확히 이해하려면, **주축(Main Axis)**과 **교차축(Cross Axis)**의 개념을 명확히 알고 있어야 한다.
Flexbox의 모든 정렬과 크기 계산은 **이 두 축을 기준**으로 이루어진다.

---

1. 주축 (Main Axis)

- **아이템이 배치되는 방향**
- `flex-direction` 속성에 의해 설정된다
- `justify-content` 속성은 이 **주축을 따라 아이템을 정렬**한다

예: `flex-direction: row` (기본값)

- 주축 → **왼쪽** → **오른쪽**
- 교차축 → 위쪽 → 아래쪽

예: `flex-direction: column`

- 주축 → **위쪽** → **아래쪽**
- 교차축 → 왼쪽 → 오른쪽

---

2. 교차축 (Cross Axis)

- 주축에 **직각으로 교차하는 축**
- `align-items`, `align-self`, `align-content` 속성은 이 **교차축을 따라 정렬**을 제어한다

---

3. 방향 정리

`flex-direction` 값	Main Axis (주축)	Cross Axis (교차축)
`row` (기본값)	왼쪽 → 오른쪽	위 → 아래
`row-reverse`	오른쪽 → 왼쪽	위 → 아래
`column`	위 → 아래	왼쪽 → 오른쪽
`column-reverse`	아래 → 위	왼쪽 → 오른쪽

---

4. 시각적 예시

예: `flex-direction: row`

```
Main Axis:  → → →
+-------------------------+
| [item1] [item2] [item3] |
|                         |  ↑
|                         |  | Cross Axis
+-------------------------+
```

예: `flex-direction: column`

```
Main Axis:
↓
↓
↓
+-----------+
| [item1]   |  → Cross Axis
| [item2]   |
| [item3]   |
+-----------+
```

---

5. 정렬 속성과 축 관계

속성	작용 축	설명
`justify-content`	주축 (Main Axis)	아이템의 **수평 또는 수직 방향 배치** 제어
`align-items`	교차축 (Cross Axis)	아이템의 **세로 또는 가로 정렬** 제어
`align-self`	교차축	개별 아이템의 교차축 정렬
`align-content`	교차축	여러 줄 배치 시 전체 라인의 정렬

---

6. 실전 팁

- Flexbox에서 **축 개념을 바꿀 수 있는 유일한 속성은** `flex-direction`이다
- **축에 따라** `justify-`**와** `align-`**이 적용되는 방향이 달라진다**
- 반응형 디자인에서 레이아웃 흐름을 유연하게 바꾸고 싶을 때,
  `flex-direction`을 `row` ↔ `column`으로 전환하면 전체 축 개념이 뒤바뀜

결론

> Flexbox의 정렬은 **축 중심 사고**로 작동한다.
> 주축(Main Axis)은 아이템이 나열되는 방향이고,
> 교차축(Cross Axis)은 그와 수직인 방향이다.
> 이 개념을 정확히 이해하면 `justify-content`와 `align-items` 같은 속성을
> 훨씬 직관적이고 유연하게 사용할 수 있다.