🧩 Flexbox Deep Dive: Understanding flex-direction

1. The Default HTML Flow

- By default, HTML lays out elements in two ways:
    - **Inline elements:** flow left to right until the line ends.
    - **Block elements:** stack vertically from top to bottom.
- This is the **normal document flow** before Flexbox gets involved.

---

2. What Happens When You Use Flexbox

- When you declare display: flex on a container:
    - All its child elements automatically align **in a single row**.
    - This happens because **the default flex-direction is row**.
- So, items appear side-by-side from **left → right**.

---

3. The Concept of Axes in Flexbox

Flexbox operates on two perpendicular axes:

Axis	Default (when flex-direction: row)	Changes when flex-direction: column
Main Axis	Horizontal (left → right)	Vertical (top → bottom)
Cross Axis	Vertical (top → bottom)	Horizontal (left → right)

Everything Flexbox does — alignment, sizing, spacing — depends on which direction is considered *main*.

---

4. The flex-direction Property

This property defines **the direction of the main axis**:

```
.container {
  display: flex;
  flex-direction: row; /* default */
}
```

Values:

- row: left → right (default)
- row-reverse: right → left
- column: top → bottom

- column-reverse: bottom → top

Changing flex-direction literally rotates how the layout "flows."

---

5. Why Main Axis vs Cross Axis Matters

When you manipulate properties like flex-basis, justify-content, or align-items, they act *along the main or cross axis*.
So if you switch direction, their behavior changes too.

---

6. flex-basis: The Size Setter

- Defines the **initial size** of a flex item **along the main axis**.
- Works like a smarter version of width or height.

Example:

```
.item {
  flex-basis: 100px;
}
```

- If flex-direction: row → flex-basis controls **width**.
- If flex-direction: column → flex-basis controls **height**.

So "basis" is always measured *along the main axis*.

---

7. The Flexbox Challenge from the Lesson

You're given a container with rainbow-colored divs:

```
.container {
  display: flex;
}
```

Goal: Make them stack vertically and increase their height.

Steps:

1. **Change the direction:**
2. .container {
3. display: flex;
4. flex-direction: column;
5. }

6. **Fix the unwanted full-width issue:**
   Instead of display: flex, use:
7. display: inline-flex;

→ makes container shrink to fit its content instead of stretching across the page.

8. **Set flex-basis for all child elements:**
   - You can't apply it on the container.
   - It must go on the children (the individual boxes).

---

8. Targeting All Child Elements

Since all child divs don't share a common class, use **combinators** and the **universal selector**:

```
.container > * {
 flex-basis: 100px;
}
```

Explanation:

- .container → parent (the Flexbox container)
- > → child combinator (selects direct children only)
- * → universal selector (selects all types of child elements)

So .container > * means "every direct child inside .container."

---

9. Putting It All Together

Final CSS:

```
.container {
 display: inline-flex;
 flex-direction: column;
}

.container > * {
 flex-basis: 100px;
}
```

✅ **Result:**

- Items now stack **top → bottom**.

- Each is **100px tall**.
- Container only takes up as much width as needed (thanks to inline-flex).

---

10. Experimenting (Because Flexbox Is Weird)

Try these out to really understand:

- Change flex-direction between row and column.
- Adjust flex-basis and watch how it affects width vs. height.
- Switch inline-flex back to flex to see full-width behavior.
- Observe what happens when there's not enough space — items shrink or wrap.

---

11. Key Takeaways

- flex-direction decides the **layout flow**.
- flex-basis defines the **size along that flow**.
- inline-flex makes the container **fit its content** instead of stretching.
- .container > * lets you style **all child items** easily.
- Flexbox is a **1D layout system** — it organizes things along one direction (row or column).

💡 **Pro Tip:**
Don't fight Flexbox. Experiment. Break it. Resize the browser.
It's one of those things that only makes sense after you've seen it misbehave a few times.