🌐 Flexbox Sizing, Shrinking & Growing – The Complete Guide

---

1. Default Behavior of Flex Items

When you put block elements (like <p> tags) inside a Flex container:

- They become **flex items** automatically.
- **Main axis** = horizontal (row) by default.
- Items sit **next to each other**, no gaps unless you add one.
- When the window shrinks, items **auto-shrink** to fit (until their *minimum content width*).

🧩 **Minimum content width:**
Determined by the **longest unbreakable word** in the item.
If that word can't fit, the box stops shrinking further.
Everything else overflows off-screen.

---

2. The Flexbox Sizing Algorithm (Priority Order)

Flexbox figures out an item's size using this order:

Priority	Property Checked	Description
1️⃣	min-width / max-width	Hard limits for shrinking or growing.
2️⃣	flex-basis	The starting size *along the main axis* (width in row, height in column).
3️⃣	width / height	Used only if flex-basis isn't set.
4️⃣	Content width	The natural size of the item based on its content.

👉 If one is set, Flexbox ignores the lower-priority ones.

---

3. Default (Auto) Sizing Example

```
.container {
  display: flex;
  gap: 10px;
}
```

- Items adjust size automatically.
- Each box shrinks until it reaches its **minimum content width** (the longest word).
- When there's more space, items stretch back to their ideal (content) width.

---

4. Setting Explicit Widths

```
.item {
  width: 100px;
}
```

- All items start at 100px.
- When container width < total items' width, Flexbox **forces them to shrink** (ignores width if necessary).
- Shrinking respects *minimum width* (longest word).

So if four items each need 100px → total = 400px.
If container < 400px → each starts shrinking, but unevenly depending on content.

---

5. Using flex-basis

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

- flex-basis sets the **starting size** of a flex item on the **main axis**.
- If set, it **overrides** the width/height property.
- Works same as width in flex-direction: row.
- If there's enough space, it takes the flex-basis size; if not, it shrinks to min-width.

💡 **Remember:** flex-basis > width.
No need to set both.

---

6. Setting max-width and min-width

Property	Function
max-width	Defines how *large* an item can grow.
min-width	Defines how *small* an item can shrink.

Example:

```
.item {
  flex-basis: 200px;
  max-width: 100px; /* smaller than flex-basis */
```

}

➡️ Actual width becomes 100px, because max-width wins.

If min-width: 300px and flex-basis: 200px,
➡️ Actual width = 300px (since min-width > flex-basis).

🧠 **Hierarchy logic:**
Flexbox always respects hard constraints (min-width, max-width) before anything else.

---

7. The Role of flex-grow and flex-shrink

These two control whether an item can **expand or contract** when container space changes.

(a) flex-grow

Defines how much an item can grow *beyond* its base size.

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

- If there's extra space, each item with flex-grow: 1 expands **equally**.
- If one has flex-grow: 2, it takes *twice the extra space* of one with grow 1.

(b) flex-shrink

Defines how much an item can **shrink** when there's *not enough* space.

```
.item {
 flex-shrink: 1; /* default */
}
```

- If flex-shrink: 0, the item **refuses to shrink** — it keeps its size.
- If flex-shrink: 2, it shrinks *twice as fast* as one with shrink 1.

(c) When Both Are Off

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

Items stay frozen at their base size — no growing, no shrinking.
Basically, Flexbox stops flexing. Static boxes.

---

8. The Magic Trio: flex-grow, flex-shrink, flex-basis

You can write all three in one go:

```
.item {
  flex: 1 1 0; /* grow shrink basis */
}
```

Or use short forms:

Shorthand	Expands To	Meaning
flex: 1	flex: 1 1 0	Grow & shrink equally, no base width.
flex: 2	flex: 2 1 0	Grows twice as much as a flex:1 item.

🧮 **Ratios:**
If one item is flex: 1 and another is flex: 2,
then total width divides as 1:2 — one-third and two-thirds respectively.

---

9. The Default Settings

By default, Flexbox behaves as if:

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

- Items do **not grow** (grow=0).
- They **can shrink** (shrink=1).
- Their base size is determined by **content** (basis=auto).

---

10. Equal Width Flex Items

If you want all boxes equal width — no matter content:

```
.item {
  flex: 1; /* shorthand for grow:1, shrink:1, basis:0 */
}
```

This makes them stretch evenly across the row, splitting space equally.

---

11. Example: Mixed Behavior

.item1 { flex: 1; } /* Grows & shrinks normally */

.item2 { flex: 2; } /* Twice the width of item1 */

.item3 { flex: 1 0 200px; } /* Starts at 200px, doesn't shrink */

Result:

- Item2 dominates space.
- Item3 keeps minimum 200px width.
- Item1 fills leftover evenly.

---

12. Recap – Flex Sizing Hierarchy

Step	Property	Description
1	min-width / max-width	Hard limits
2	flex-basis	Main-axis size
3	width / height	Used only if no basis
4	Content size	Natural size
5	flex-grow / flex-shrink	Adjust dynamically to available space

---

13. Common Use Cases

Goal	CSS Rule
Equal boxes	flex: 1;
Fixed item, others flexible	flex: 0 1 auto;
One item double width	flex: 2;
Prevent shrinking	flex-shrink: 0;
Set min & max limits	min-width, max-width

---

14. Example Visual Summary

Property	Expands?	Shrinks?	Base Size
flex: 0 0 100px;	❌	❌	100px
flex: 1 0 100px;	✅	❌	100px
flex: 0 1 100px;	❌	✅	100px
flex: 1 1 0;	✅	✅	Equal width
flex: 2 1 0;	✅ ✅	✅	Double share