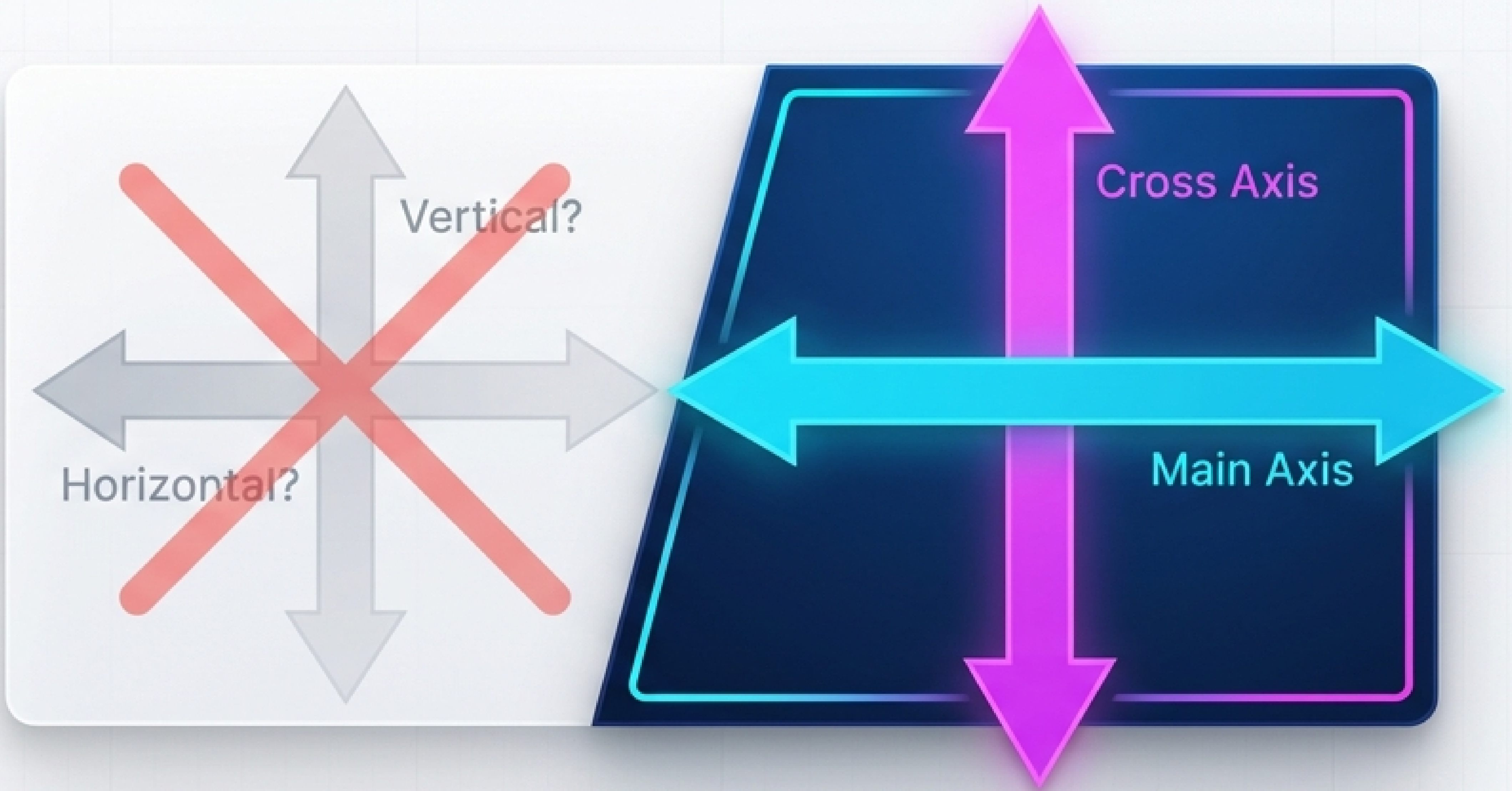
Flexbox: A New Mental Model for Layout

NotebookLM

Know thine axes.

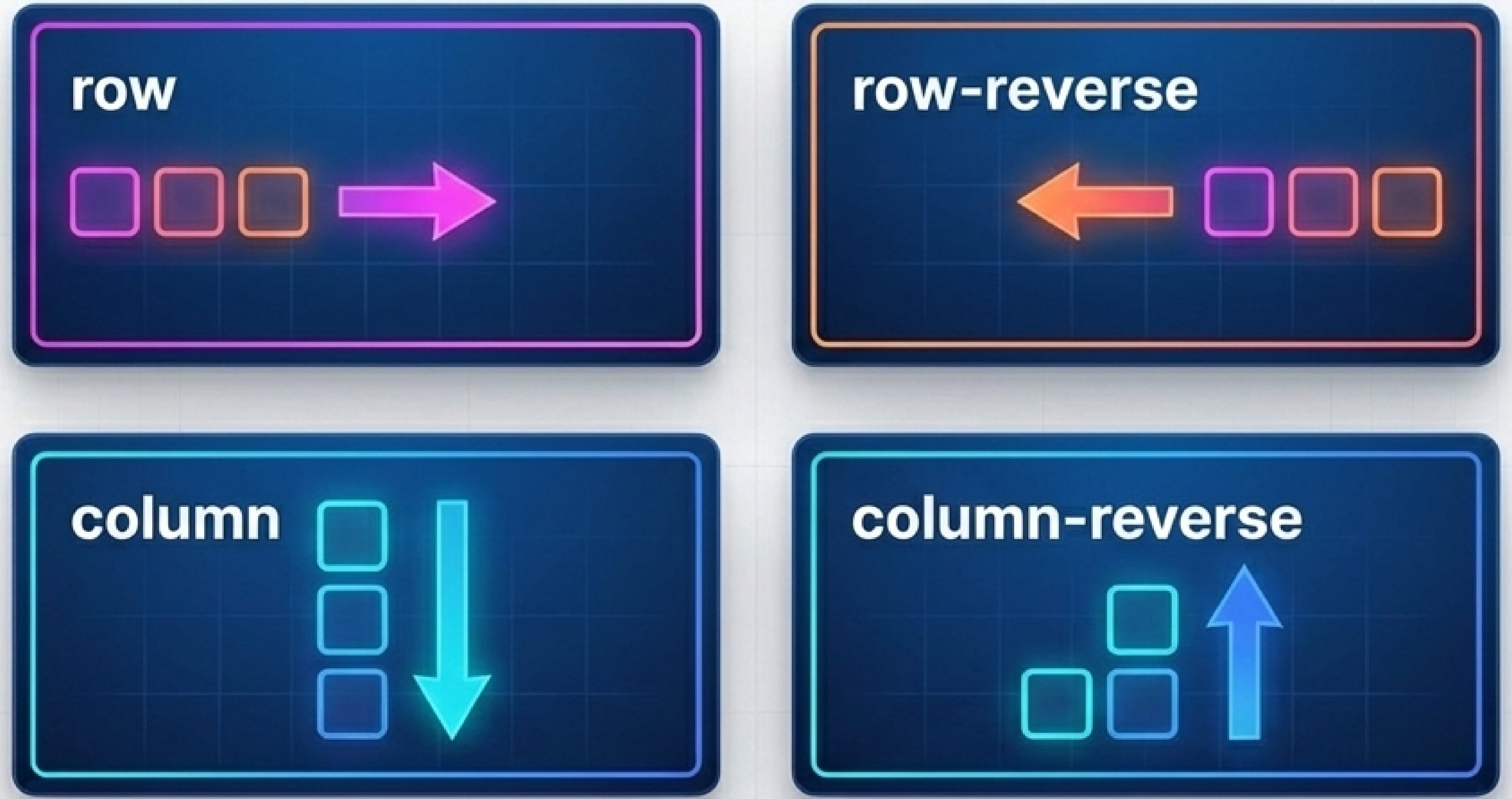
Flexbox doesn’t think in horizontal and vertical — it thinks in **main axis** and **cross axis**.



And the single property that defines both is `flex-direction`.

`flex-direction` is the steering wheel. The axes are the roads.

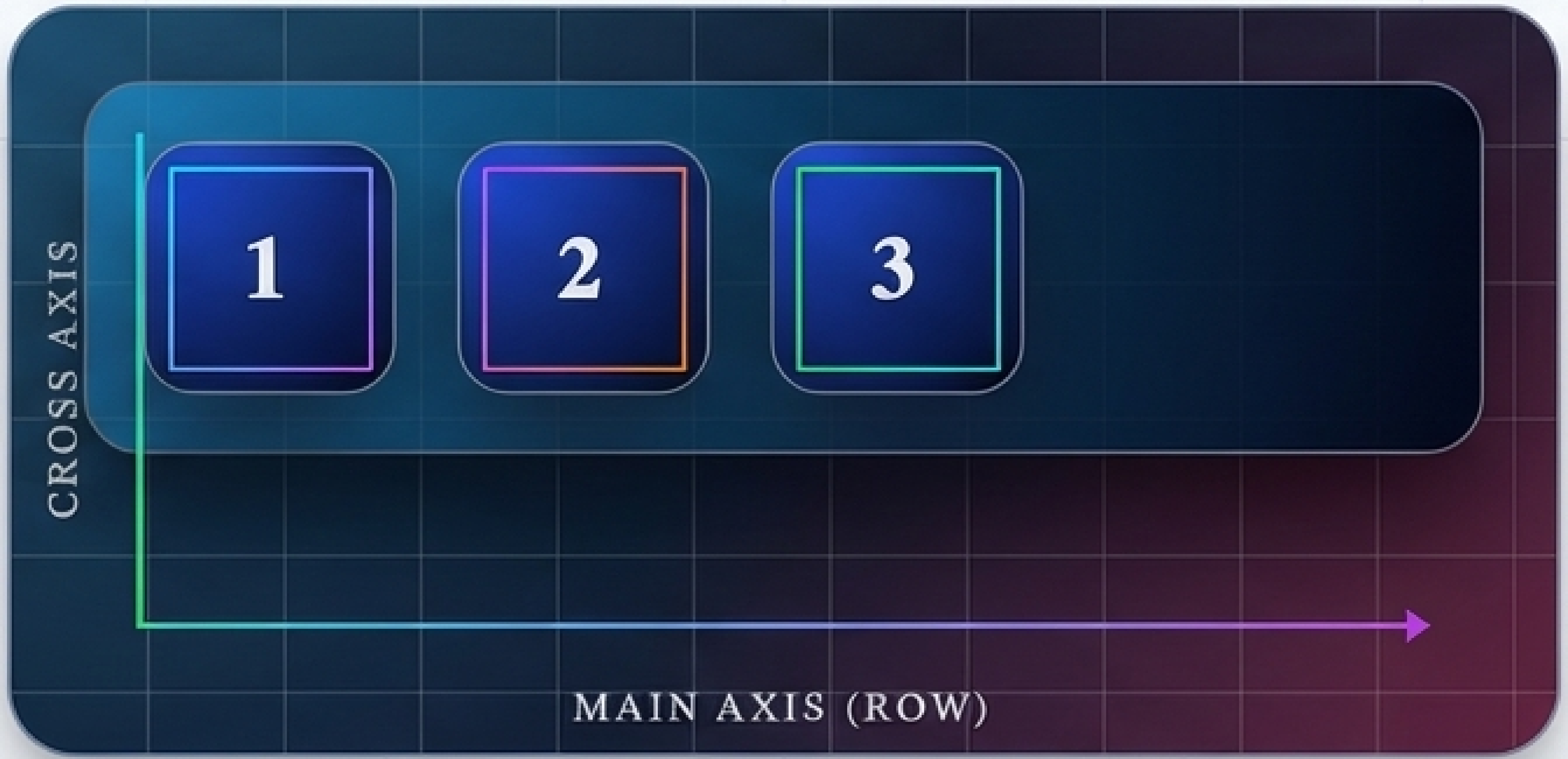
This one property determines how items flow, where the main and cross axes run, and how all alignment, spacing, and wrapping will behave.



NotebookLM

`Row”: the main axis runs left → right

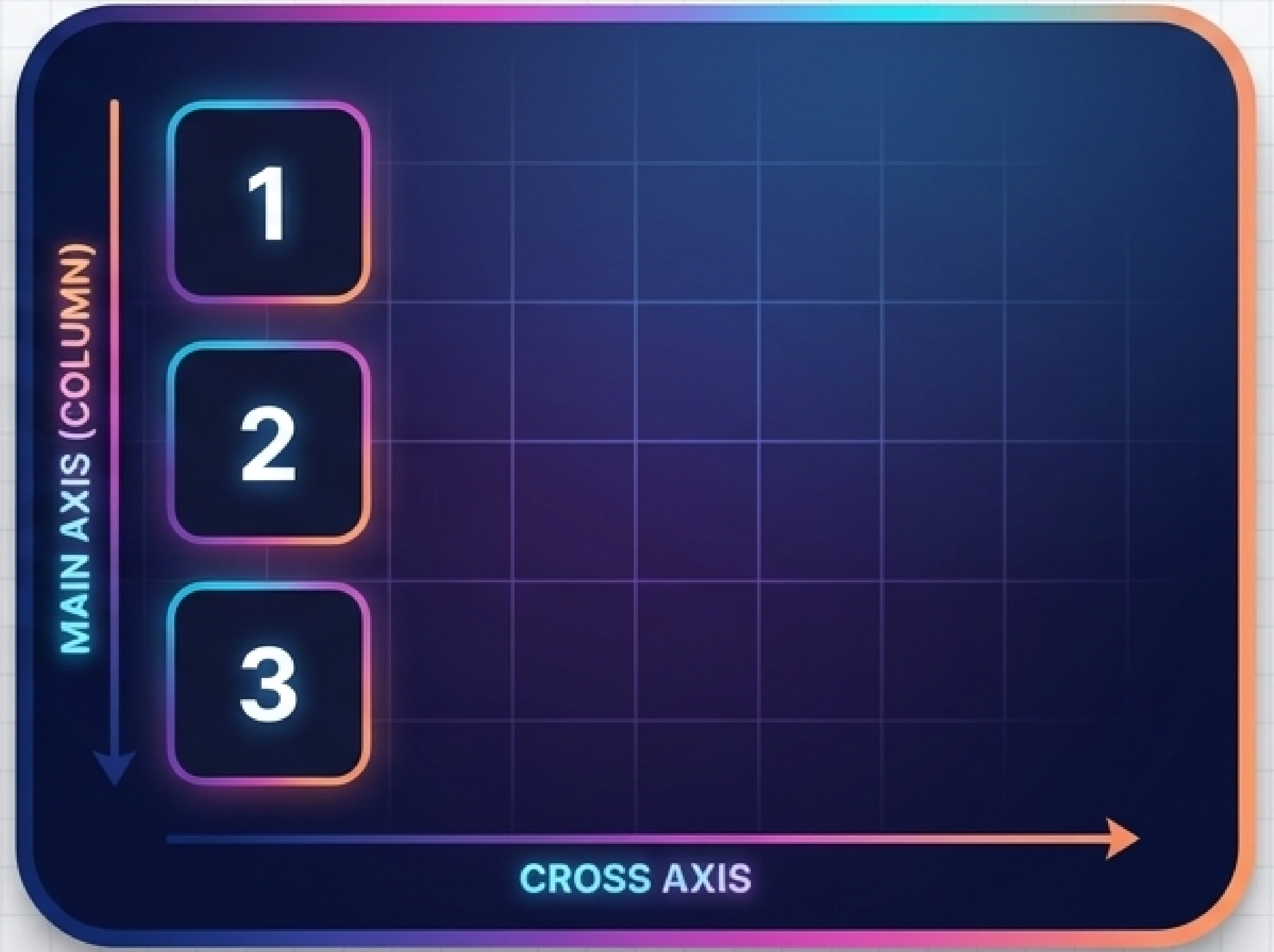
The default direction is `row`. The main axis runs horizontally, and the items walk from left to right in source order. The cross axis is vertical.



NotebookLM

`Column`: the main axis runs top → bottom

With **`flex-direction: column`**, the main axis rotates. The items now march vertically, but the cross axis becomes the horizontal.



The Golden Rule of Flexbox Alignment

Every flexbox power move depends on knowing the axes.
The rule is simple and absolute:

`justify` = main axis ↔

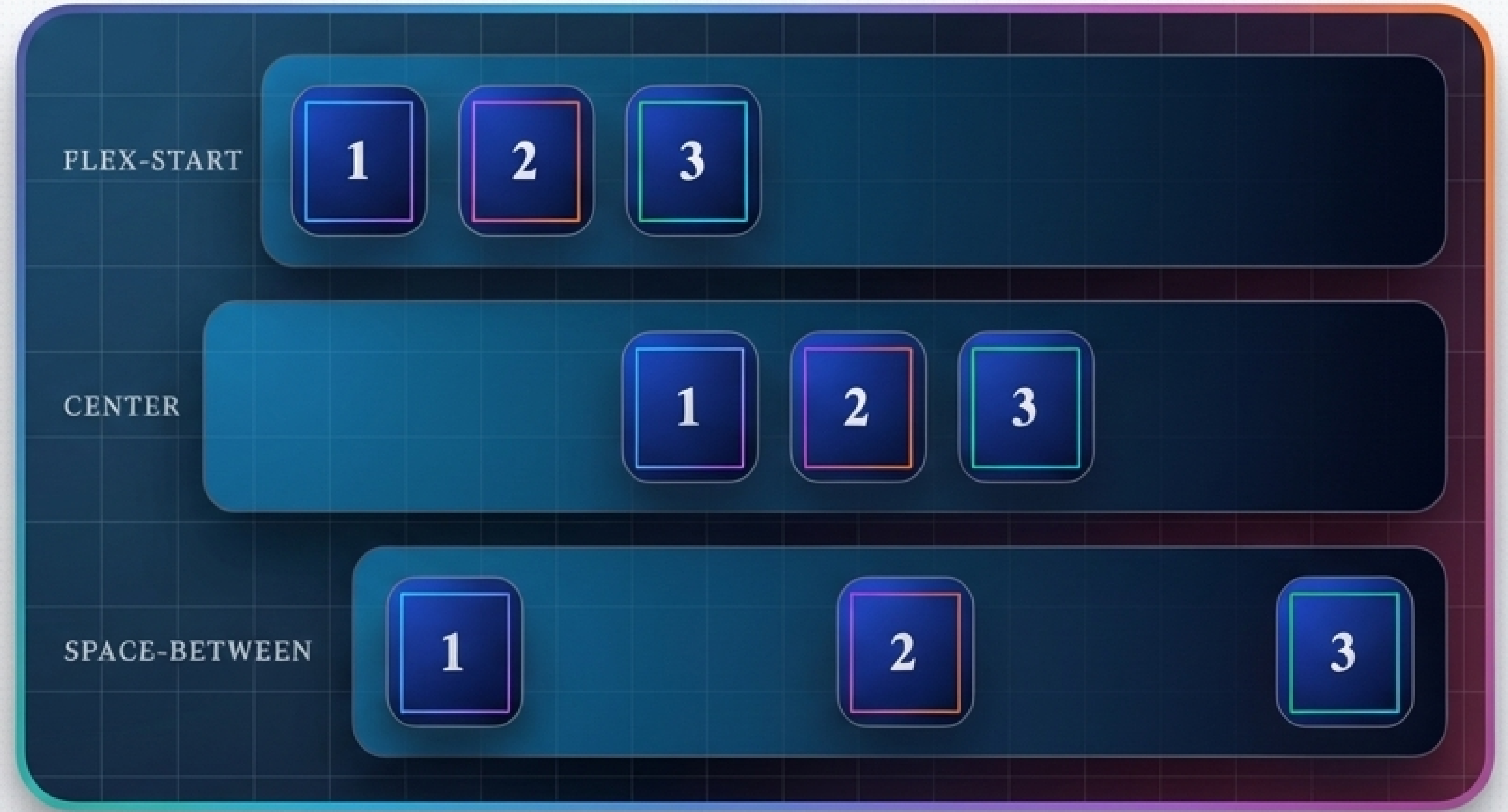
`align` = cross axis ↕

Now that our map is clear, we can place items with precision.

Justify: how items spread on the main axis

`justify-content` distributes available space along the main axis. With a `row`' direction, this moves items along the horizontal axis: packed at the start, centered, or spaced out.

- `flex-start` (default)
- `center`
- `flex-end`
- `space-between`
- `space-around`
- `space-evenly`

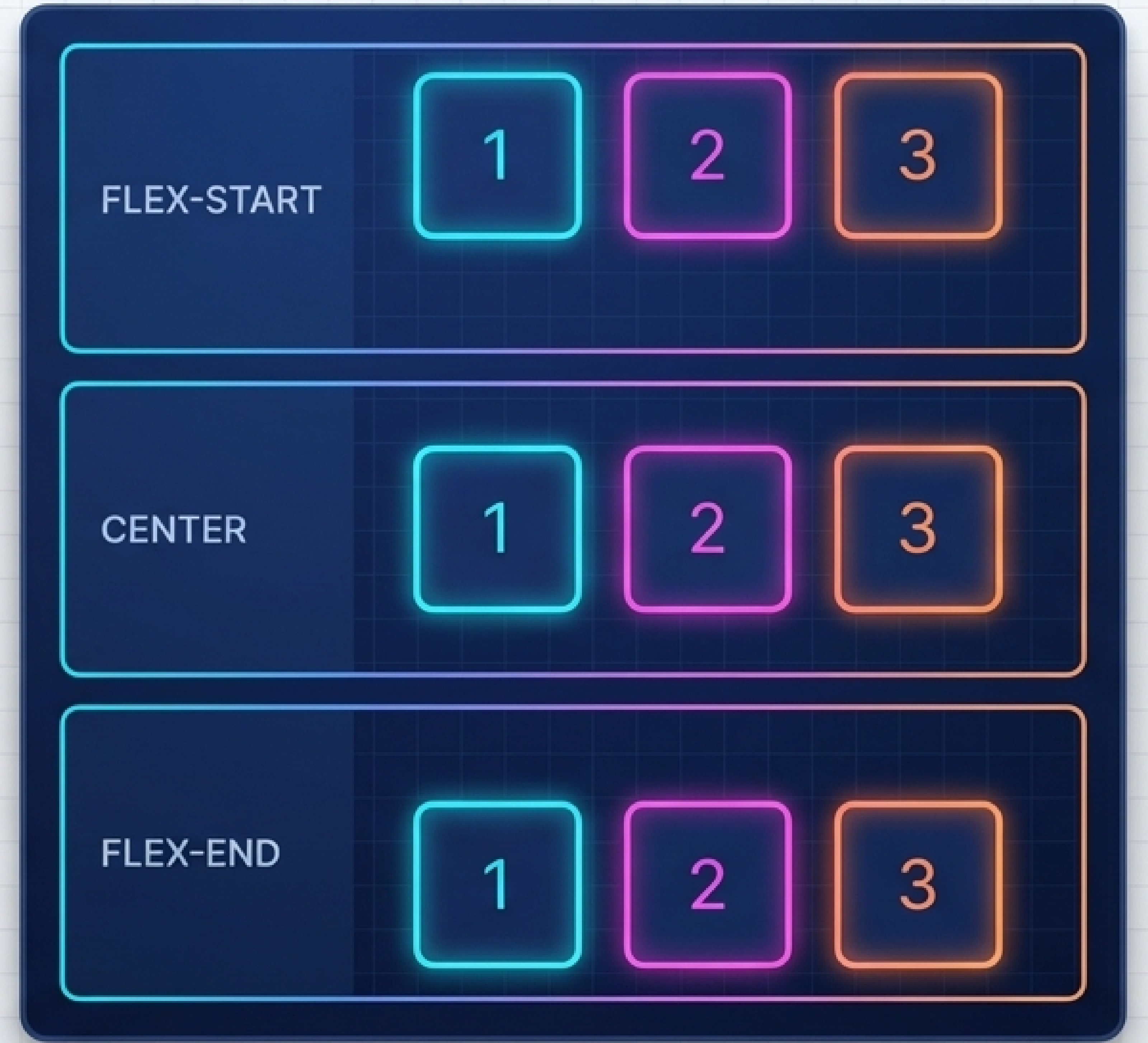


Align: how items rest on the cross axis

`align-items` positions items along the cross axis—always perpendicular to the main flow. With a `row` direction, this slides the items top, middle, or bottom inside a taller container.

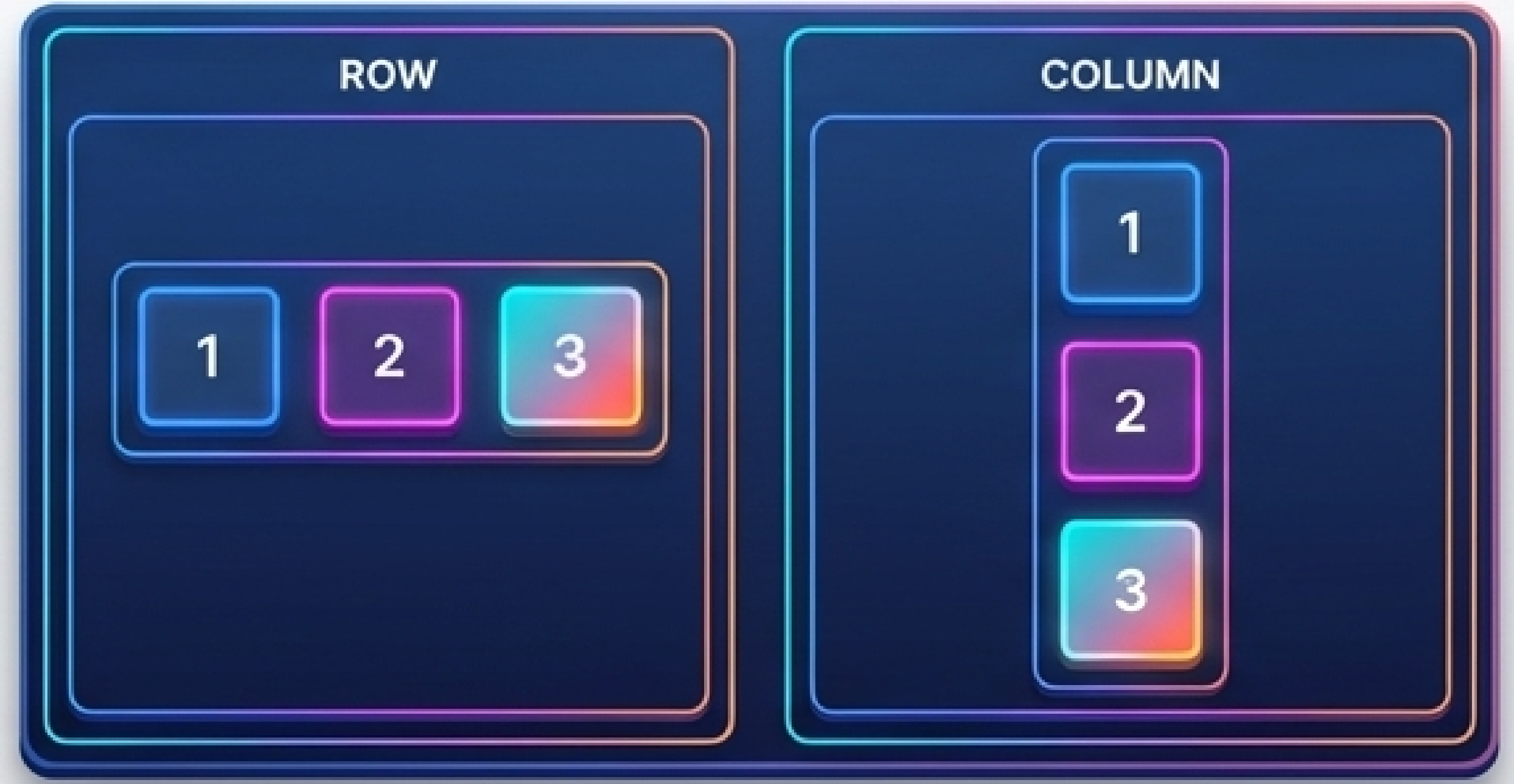
- `stretch` (default)
- `flex-start`
- `center`
- `flex-end`

ⓘ Note: `stretch` is the quiet default. Items will fill the cross axis unless told otherwise.

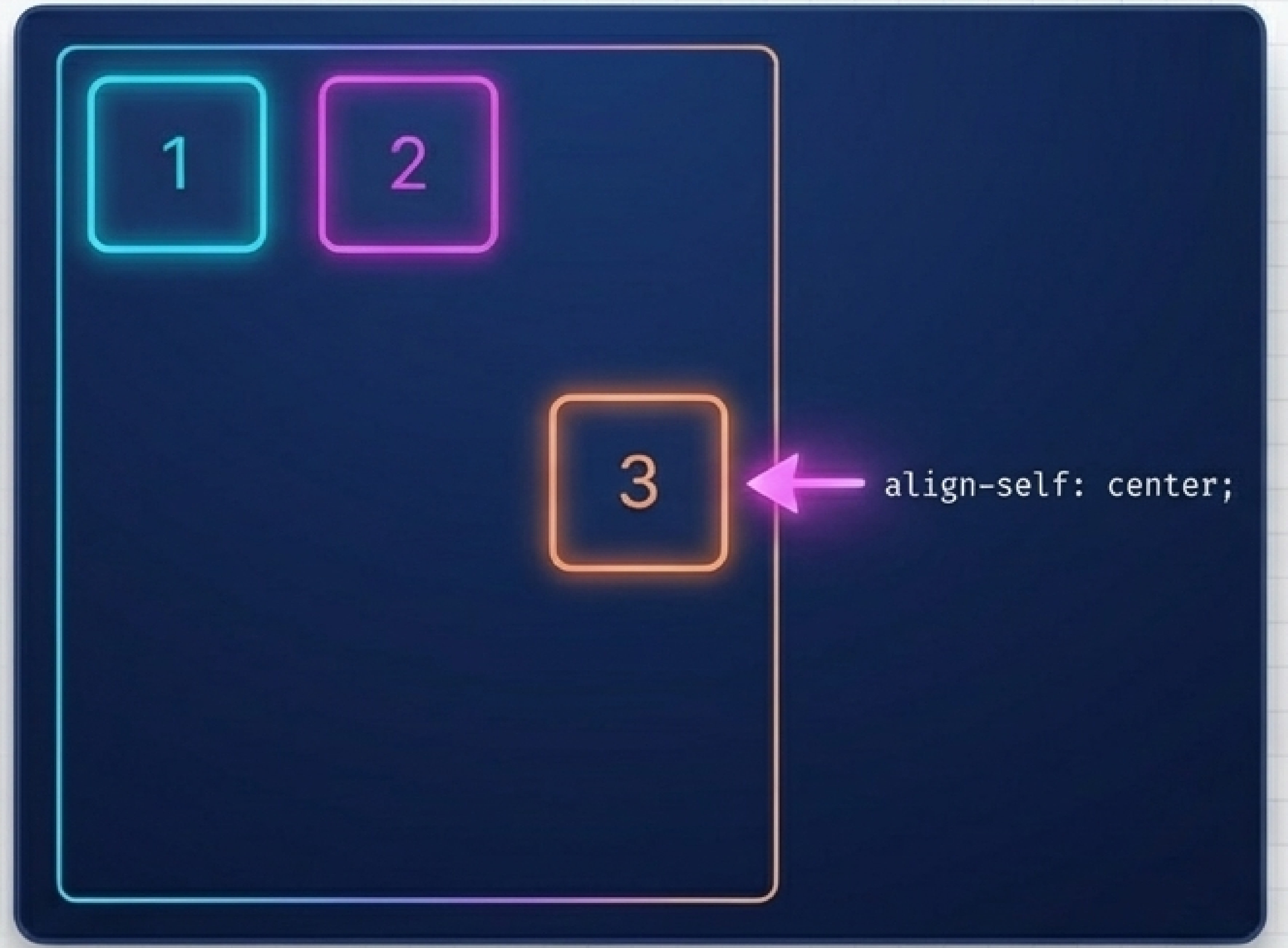


Swap the axis: same words, new directions

When you change `flex-direction` to `column`, the main axis rotates. The mental model holds: `justify-content` still follows the main axis (now top-to-bottom), and `align-items` still follows the cross axis (now left-to-right).



Swap directions, swap axes, keep your sanity.



When one item stands apart

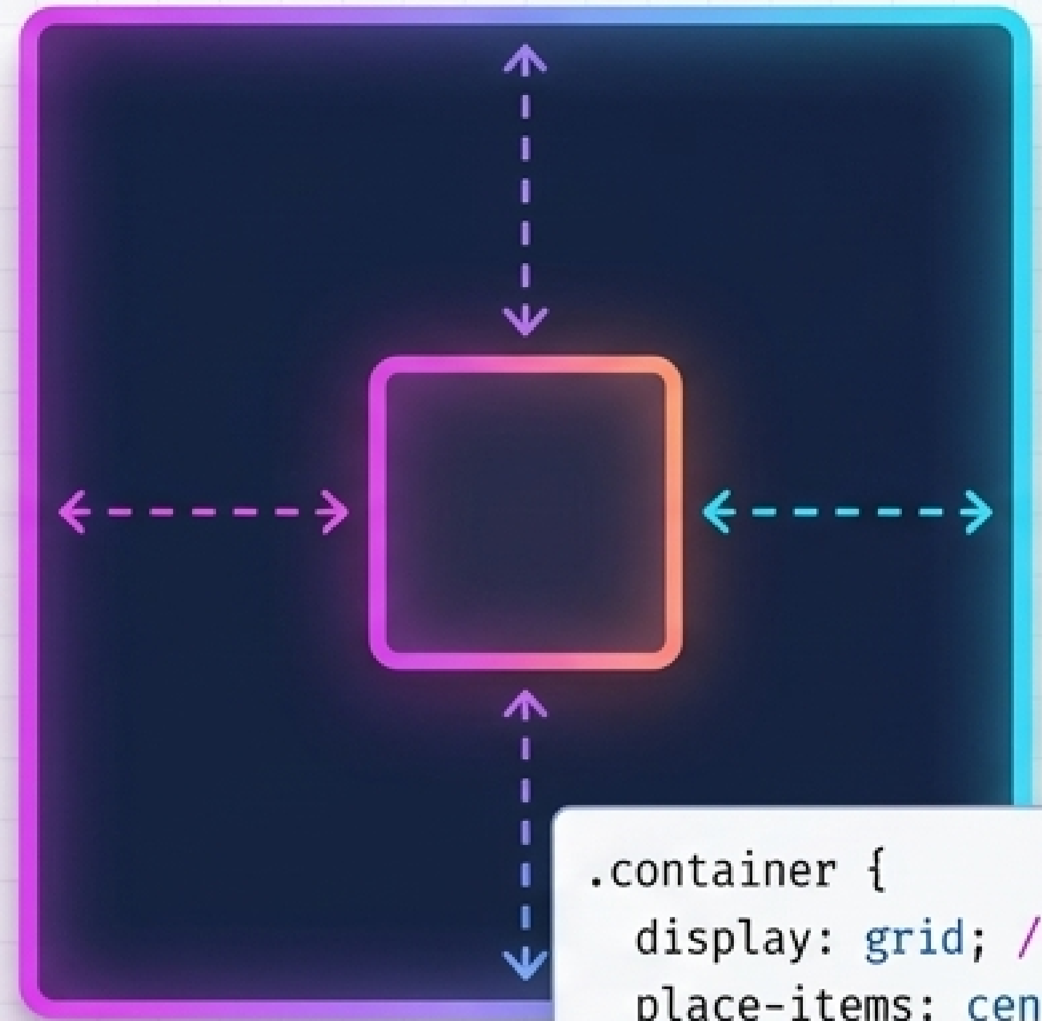
Group alignment is harmonious... until one item wants special treatment. `align-self` overrides `align-items` for a single element, letting it rise, sink, or center across the cross axis without disturbing its neighbors.

```css
.item-3 {
  align-self: center;
}
```

NotebookLM

The Centralizing Power Move

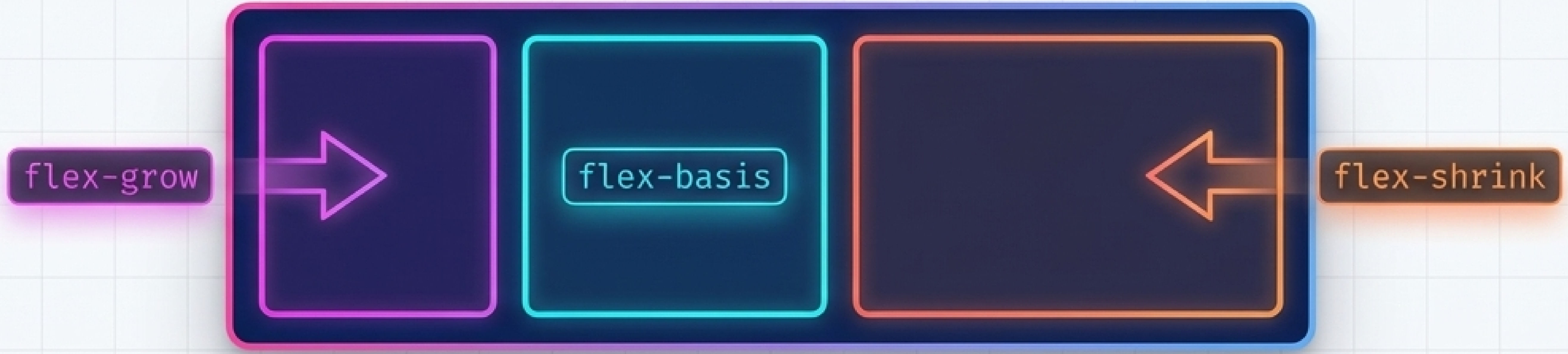
While not fully applicable to Flexbox containers (`justify-items` doesn't apply), the `place-items` property endures as a one-line mental model for "put it smack in the middle."



```css
.container {
  display: grid; /* Works perfectly in Grid! */
  place-items: center;
}
```

Next Step: Space Negotiation

With direction and alignment under our belt, we move into the dynamic heart of Flexbox: `flex-grow`, `flex-shrink`, and `flex-basis`. This is how items bargain for space along the main axis, and who expands, contracts, or holds their ground.


flex-grow
flex-basis
flex-shrink