|  |  |
| --- | --- |
| Syntax | |
| /\* Comments \*/ *Note:*  @media type { *media type is optional*  selector {  property: value;  }  } | |
| Inline Style  <tag style="property: value;"> | |
| Embedded Style  <head>  <style type="text/css">  selector { property: value; }  </style>  </head> | |
| External Style Sheet  <head>  <link rel="stylesheet"  padding-top padding-right  padding-bottom padding-left  padding  margin-top margin-right  margin-bottom margin-left  margin  Boxes  type="text/css" href="style.css" />  </head> | |
| Selectors | |
| \* | All elements |
| tag | All tag elements |
| tag \* | All elements within tag |
| tag tag2 | border-top-width border-right-width  border-bottom-width border-left-width  border-width  border-top-style border-right-style  border-bottom-style border-left-style  border-style  border-top-color border-right-color  border-bottom-color border-left-color  border-color  border-top border-right  border-bottom border-left  border  tag2 elements (any childhood level ) within tag |
| tag, tag2 | All tag and tag2 elements |
| tag > tag2 | tag2 is a direct child of tag |
| tag + tag2 | tag2 preceded by tag |
| .class | Elements with class ‘class’ |
| tag.class | All tags with class ‘class’ |
| #id | Element with id ‘id’ |
| tag#id | Tag with id ‘id’ |

Box Model

|  |  |  |
| --- | --- | --- |
| Size and colors | | |
| Relative sizes | em | rem | ex | ch  vw | vh | vmin | vmax | % | |
| Absolute sizes | cm | mm | in | px | pt | pc | |
| Colors | Hex #ff00ff  RGB rgb(255,0,255) | |
| Positioning | | |
| position | absolute | fixed | relative | static | inherit | |
| float | left | right | none | inherit | |
| top, right,  bottom, left | Sets the offset from the edge | |
| clear | left | right | both | none | inherit I fixed the document now | |
| display | Specifies element placing in the document flow | |
| overflow | visible | hidden | scroll | auto | inherit | |
| visibility | visible | hidden | collapse | inherit | |
| z-index | auto | int | inherit | |
| Dimensions | | |
| height | | auto | int | % | inherit |
| max-height | | none | int | % | inherit |
| max-height | | none | int | % | inherit |
| max-width | | none | int | % | inherit |
| min-height | | int | % | inherit |
| min-width | | int | % | inherit |
| width | | auto | int | % | inherit |
| Other | | |
| background | | Format:  background (color) (image) (repeat) (position) |
| cursor | | Set the type of cursor |
| quotes | | Set type of quotation marks |

|  |  |
| --- | --- |
| Pseudo-selectors | |
| :active | Adds style to active element |
| :after | Adds content after element |
| :before | Ads content before element |
| : first-child | Adds style to first child |
| : first-letter | Adds style to first character |
| : first-line | Adds style to first line |
| :focus | Adds style to focused element |
| :hover | Adds style when mouse is over |
| :link | Adds style to unvisited link |
| :visited | Adds style to visited link |
| Text | |
| font-family | Specifies the font family |
| font-size | Specifies the font size |
| font-style | normal | italic | oblique | inherit |
| font-variant | normal | small-caps | inherit |
| font-weight | normal | bold | bolder | lighter | int (100- 900) | inherit |
| color | Sets the color of text |
| line-height | normal | int | % | inherit |
| text-align | left | right | center | justify | inherit |
| text-  decoration | none | underline | overline | line-through | blink | inherit |
| text-indent | int | % | inherit |
| text-  transform | none | capitalize | uppercase | lowercase | inherit |
| vertical  -align | int | % | baseline | sub | super | top | text-top  middle | bottom | text-bot- tom | inherit |
| white  -space | normal | nowrap | pre |  pre-line | pre-wrap| inherit |
| word-  spacing | normal | length | inherit |

**CSS Precedence Rules**

When the browser needs to resolve what styles to apply to a given HTML element, it uses a set of CSS precedence rules. Given these rules, the browser can determine what styles to apply. The rules are:

1. !important after CSS properties.
2. Specificity of CSS rule selectors.
3. Sequence of declaration.

Note, that CSS precedence happens at CSS property level. Thus, if two CSS rules target the same HTML element, and the first CSS rule takes precedence over the second, then all CSS properties specified in the first CSS rule takes precedence over the CSS properties declared in the second rule. However, if the second CSS rule contains CSS properties that are not specified in the first CSS rule, then these are still applied. The CSS rules are combined - not overriding each other.

The different CSS selector types has different specificity. By specificity is meant how specifically the CSS selector targets the element is selects. Here is a list of CSS selector specificity:

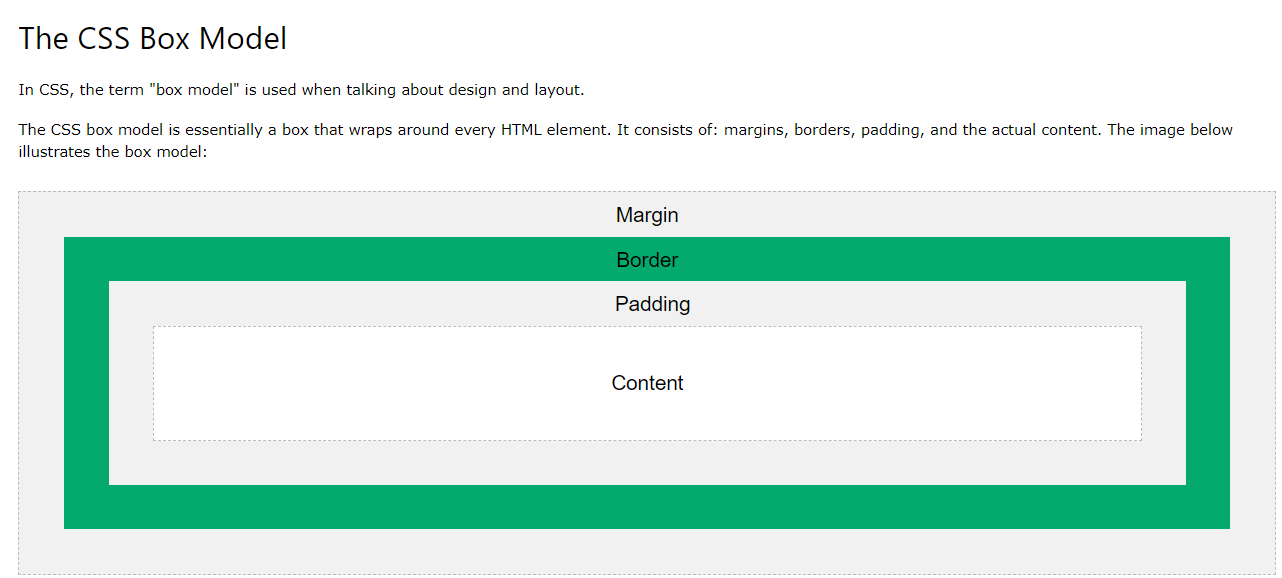
|  |  |
| --- | --- |
| **CSS Selector** | **Description** |
| Inherited styles | Lowest specificity of all selectors - since an inherited style targets the element's parent, and not the HTML element itself. |
| \* | Lowest specificity of all directly targeted selectors |
| element | Higher specificity than universal selector and inherited styles. |
| attribute | Higher specificity than element selector |
| Class | Higher specificity than attribute, element and universal selectors. |
| ID | Higher specificity than class selector. |
| Combined selectors | Gets the specificity of the selectors combined. |
| CSS properties set directly on element, inside style attribute. | Stronger specificity than ID selector. |

CSS selectors:

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **selects** |
| [.*class*](https://www.w3schools.com/cssref/sel_class.asp) | .intro | all tags with class="intro" |
| .class1.class2 | .name1.name2 | all tags with both name1 and name2 set within its class attribute |
| .class1 .class2 | .name1 .name2 | all tags with name2 that is a descendant of an tag with name1 |
| [#*id*](https://www.w3schools.com/cssref/sel_id.asp) | #firstname | the tag with id="firstname" |
| [\*](https://www.w3schools.com/cssref/sel_all.asp) | \* | all tags |
| [*tag*](https://www.w3schools.com/cssref/sel_element.asp) | p | all <p> tags |
| [*tag.class*](https://www.w3schools.com/cssref/sel_element_class.asp) | p.intro | all <p> tags with class="intro" |
| [*tag,tag*](https://www.w3schools.com/cssref/sel_element_comma.asp) | div, p | all <div> tags and all <p> tags |
| [*tag* *tag*](https://www.w3schools.com/cssref/sel_element_element.asp) | div p | all <p> tags inside <div> tags |
| [*tag*>*tag*](https://www.w3schools.com/cssref/sel_element_gt.asp) | div > p | all <p> tags where the parent is a <div> tag |
| [*tag*+*tag*](https://www.w3schools.com/cssref/sel_element_pluss.asp) | div + p | the first <p> tag that is placed immediately after <div> tags |
| [*tag1*~*tag2*](https://www.w3schools.com/cssref/sel_gen_sibling.asp) | p ~ ul | every <ul> tag that is preceded by a <p> tag |
| [[*attribute*]](https://www.w3schools.com/cssref/sel_attribute.asp) | [target] | all tags with a target attribute |
| [[*attribute*=*value*]](https://www.w3schools.com/cssref/sel_attribute_value.asp) | [target=\_blank] | all tags with target="\_blank" |
| [[*attribute*~=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_contains.asp) | [title~=flower] | all tags with a title attribute containing the word "flower" |
| [[*attribute*|=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_lang.asp) | [lang|=en] | all tags with a lang attribute value starting with "en" |
| [[*attribute*^=*value*]](https://www.w3schools.com/cssref/sel_attr_begin.asp) | a[href^="https"] | every <a> tag whose href attribute value begins with "https" |
| [[*attribute*$=*value*]](https://www.w3schools.com/cssref/sel_attr_end.asp) | a[href$=".pdf"] | every <a> tag whose href attribute value ends with ".pdf" |
| [[*attribute*\*=*value*]](https://www.w3schools.com/cssref/sel_attr_contain.asp) | a[href\*="w3schools"] | every <a> tag whose href attribute value contains the substring "w3schools" |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | the active link |
| [::after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Insert something after the content of each <p> tag |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | p::before | Insert something before the content of each <p> tag |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | every checked <input> tag |
| [:default](https://www.w3schools.com/cssref/sel_default.asp) | input:default | the default <input> tag |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | every disabled <input> tag |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | every <p> tag that has no children (including text nodes) |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | every enabled <input> tag |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | every <p> tag that is the first child of its parent |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | the first letter of every <p> tag |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | the first line of every <p> tag |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | every <p> tag that is the first <p> tag of its parent |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | the input tag which has focus |
| [:fullscreen](https://www.w3schools.com/cssref/sel_fullscreen.asp) | :fullscreen | the tag that is in full-screen mode |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | links on mouse over |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | input tags with a value within a specified range |
| [:indeterminate](https://www.w3schools.com/cssref/sel_indeterminate.asp) | input:indeterminate | input tags that are in an indeterminate state |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | all input tags with an invalid value |
| [:lang(*language*)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | every <p> tag with a lang attribute equal to "it" (Italian) |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | every <p> tag that is the last child of its parent |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | every <p> tag that is the last <p> tag of its parent |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | all unvisited links |
| [::marker](https://www.w3schools.com/cssref/sel_marker.asp) | ::marker | the markers of list items |
| [:not(*selector*)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | every tag that is not a <p> tag |
| [:nth-child(*n*)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | every <p> tag that is the second child of its parent |
| [:nth-last-child(*n*)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | every <p> tag that is the second child of its parent, counting from the last child |
| [:nth-last-of-type(*n*)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | every <p> tag that is the second <p> tag of its parent, counting from the last child |
| [:nth-of-type(*n*)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | every <p> tag that is the second <p> tag of its parent |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | every <p> tag that is the only <p> tag of its parent |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | every <p> tag that is the only child of its parent |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | input tags with no "required" attribute |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | input tags with a value outside a specified range |
| [::placeholder](https://www.w3schools.com/cssref/sel_placeholder.asp) | input::placeholder | input tags with the "placeholder" attribute specified |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | input tags with the "readonly" attribute specified |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | input tags with the "readonly" attribute NOT specified |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | input tags with the "required" attribute specified |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | :root | the document's root tag |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | ::selection | the portion of an tag that is selected by a user |
| [:target](https://www.w3schools.com/cssref/sel_target.asp) | #news:target | the current active #news tag (clicked on a URL containing that anchor name) |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | all input tags with a valid value |

Example on ::after selector





**Font styling:**

When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight, In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.



Font Pairing Rules

1. Compliment 🡪 It is always safe to find font pairings that complement one another.

2. Use Font Superfamilies 🡪 A font superfamily is a set of fonts designed to work well together. So, using different fonts within the same superfamily is safe.

3. Contrast is King 🡪 Two fonts that are too similar will often conflict. However, contrasts, done the right way, brings out the best in each font.

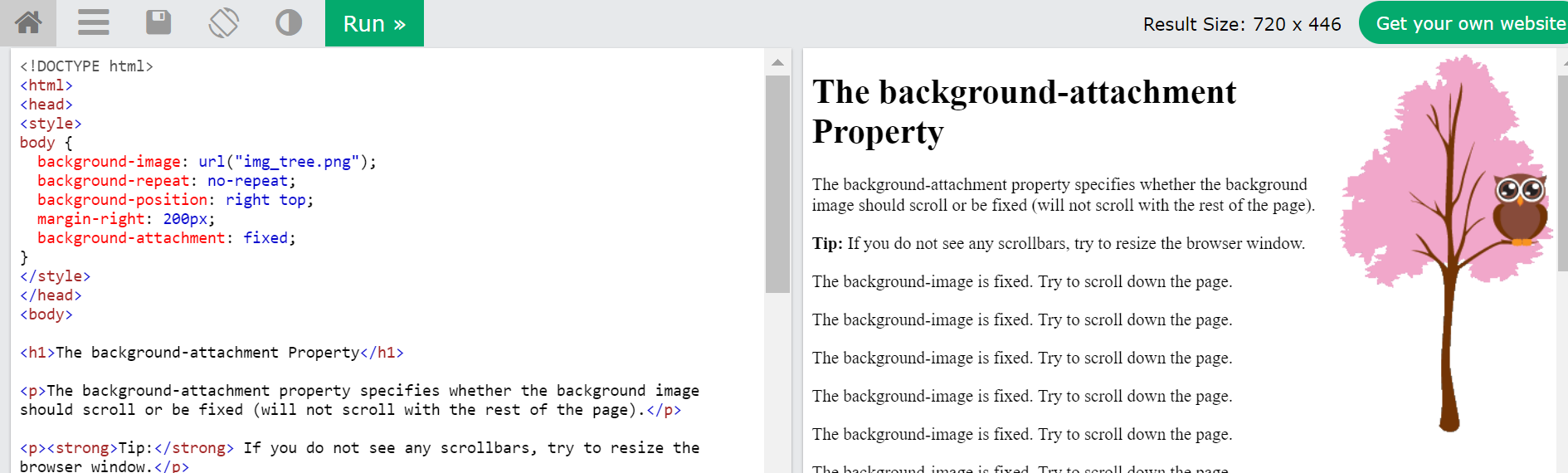
4. Choose Only One Boss

## **Block-level Elements:** A block-level element always starts on a new line and takes up the full width available, <div> , <h1> - <h6> , <p> , <form>

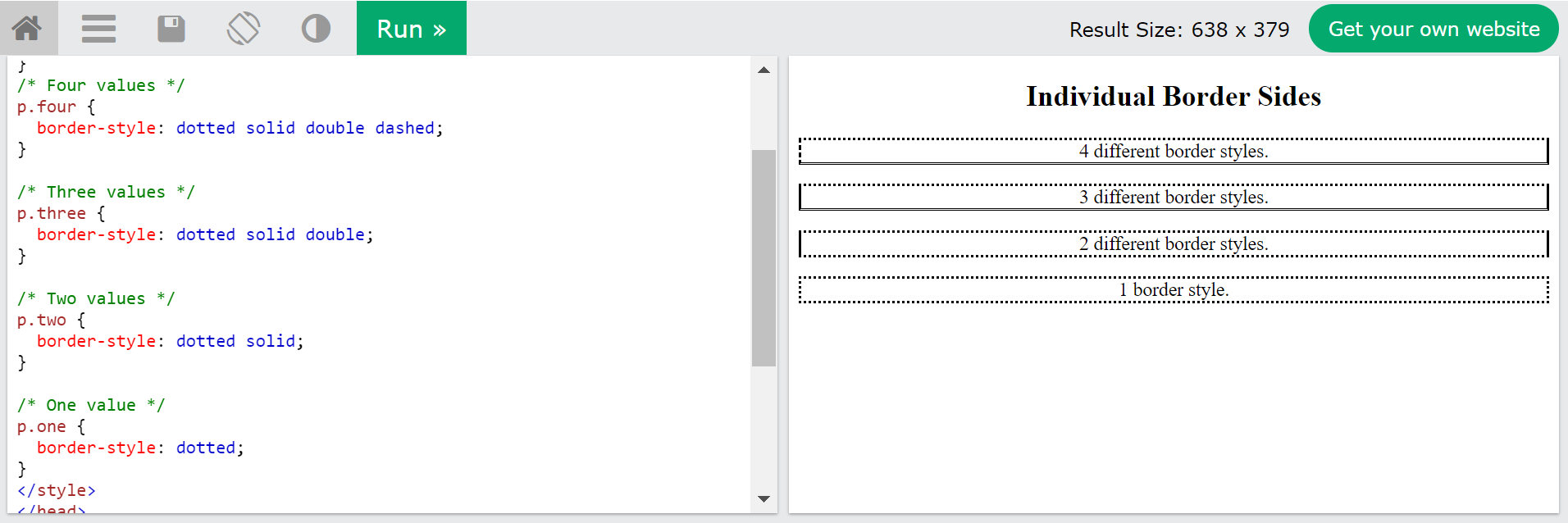
## **Inline Elements:** An inline element does not start on a new line and only takes up as much width as necessary. <span> , <a> , <img>

**Styling examples**

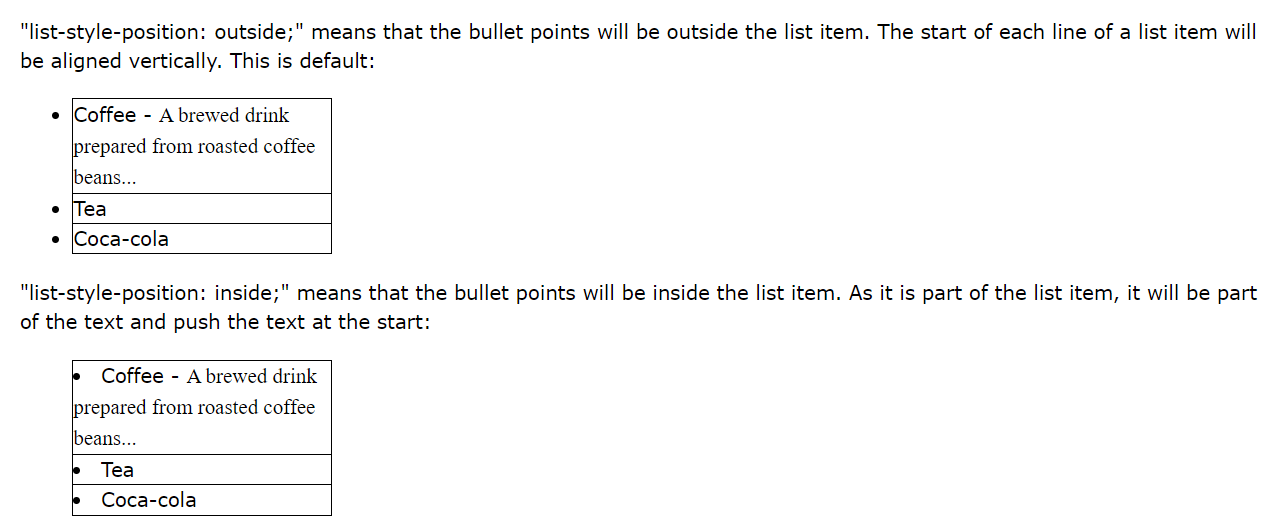
Defining fixed background:

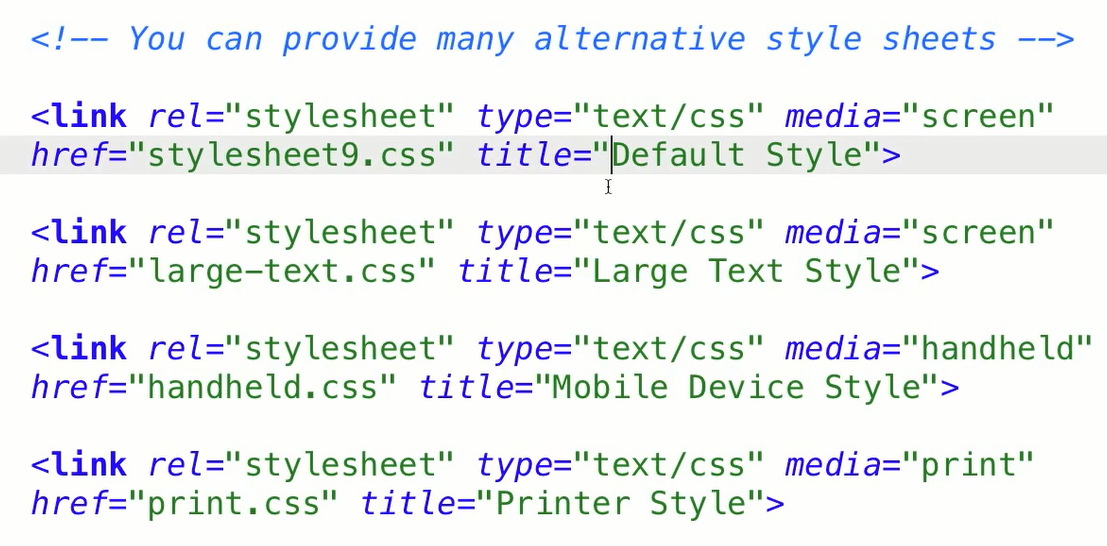


Border sides:

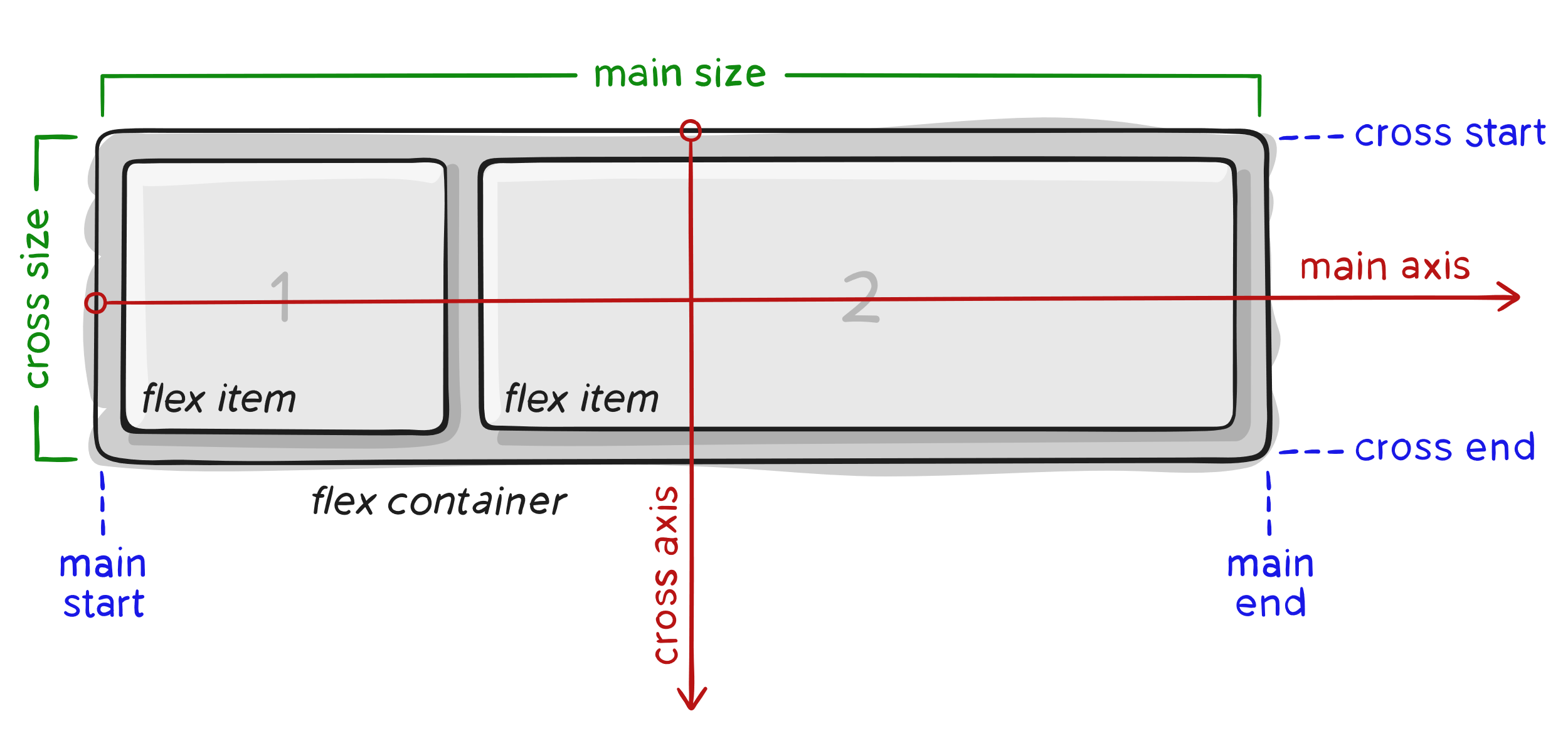


list-style-position:





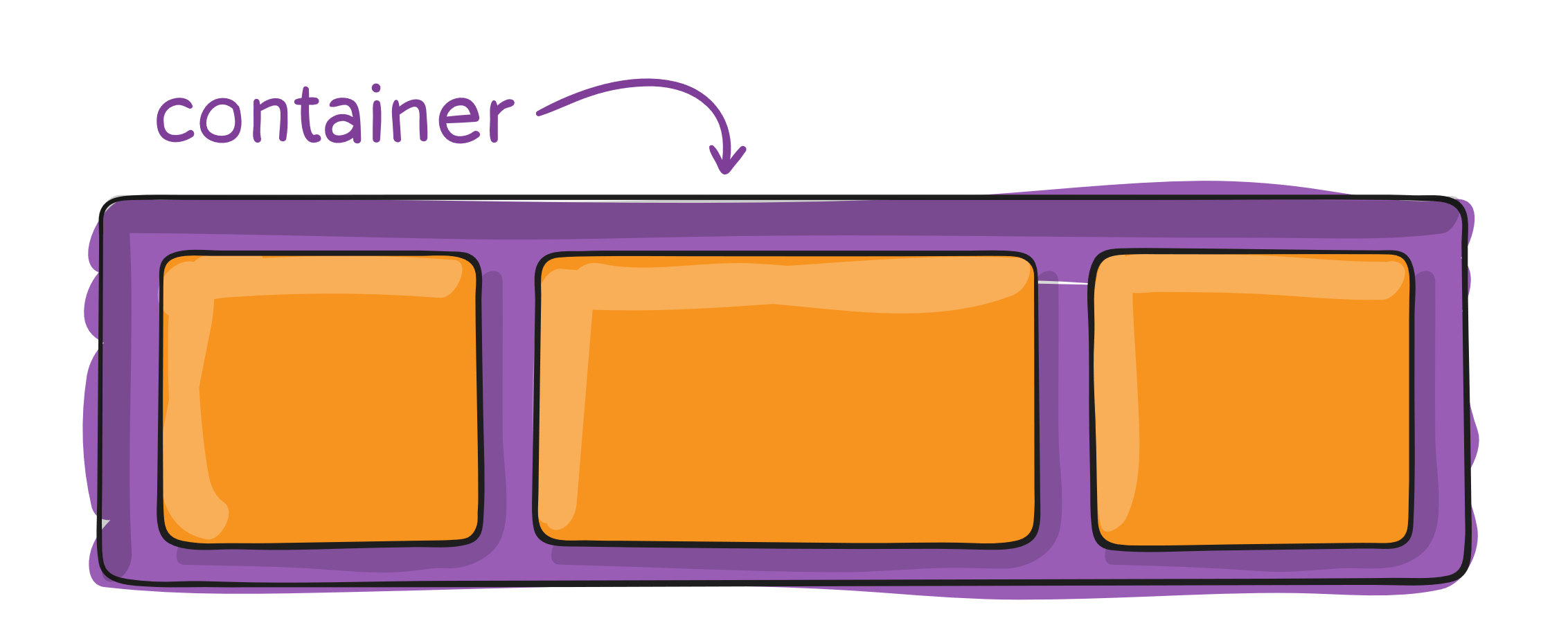
**Flexbox layout:**



Items will be laid out following either the main axis (from main-start to main-end) or the cross axis (from cross-start to cross-end).

* **main axis** – The main axis of a flex container is the primary axis along which flex items are laid out. Beware, it is not necessarily horizontal; it depends on the flex-direction property (see below).
* **main-start | main-end** – The flex items are placed within the container starting from main-start and going to main-end.
* **main size** – A flex item’s width or height, whichever is in the main dimension, is the item’s main size. The flex item’s main size property is either the ‘width’ or ‘height’ property, whichever is in the main dimension.
* **cross axis** – The axis perpendicular to the main axis is called the cross axis. Its direction depends on the main axis direction.
* **cross-start | cross-end** – Flex lines are filled with items and placed into the container starting on the cross-start side of the flex container and going toward the cross-end side.
* **cross size** – The width or height of a flex item, whichever is in the cross dimension, is the item’s cross size. The cross size property is whichever of ‘width’ or ‘height’ that is in the cross dimension.

### Flexbox properties



## Properties for the Parent (flex container)

#### display

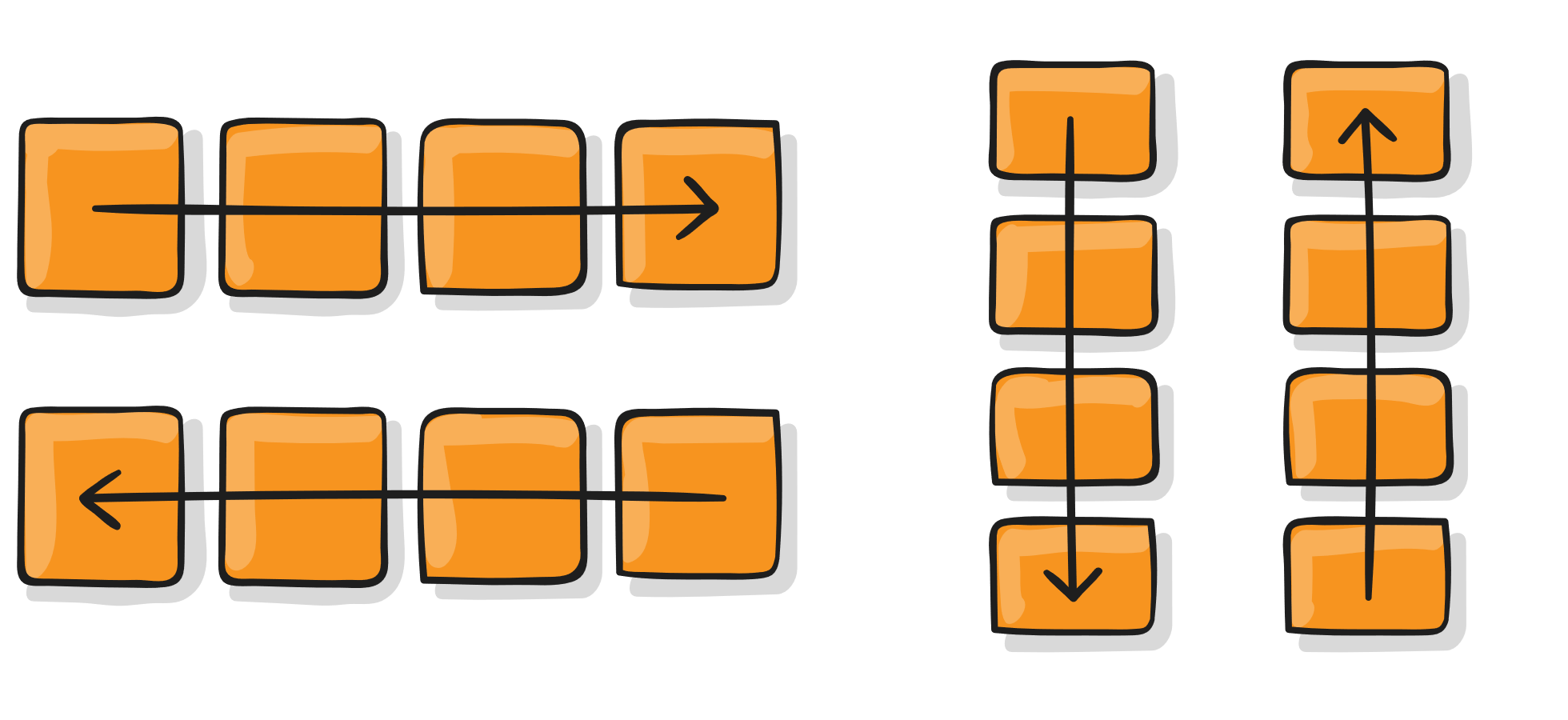
This defines a flex container; inline or block depending on the given value. It enables a flex context for all its direct children. Note that CSS columns have no effect on a flex container.

.container {

display: flex; /\* or inline-flex \*/}

#### flex-direction

Think of flex items as primarily laying out either in horizontal rows or vertical columns.

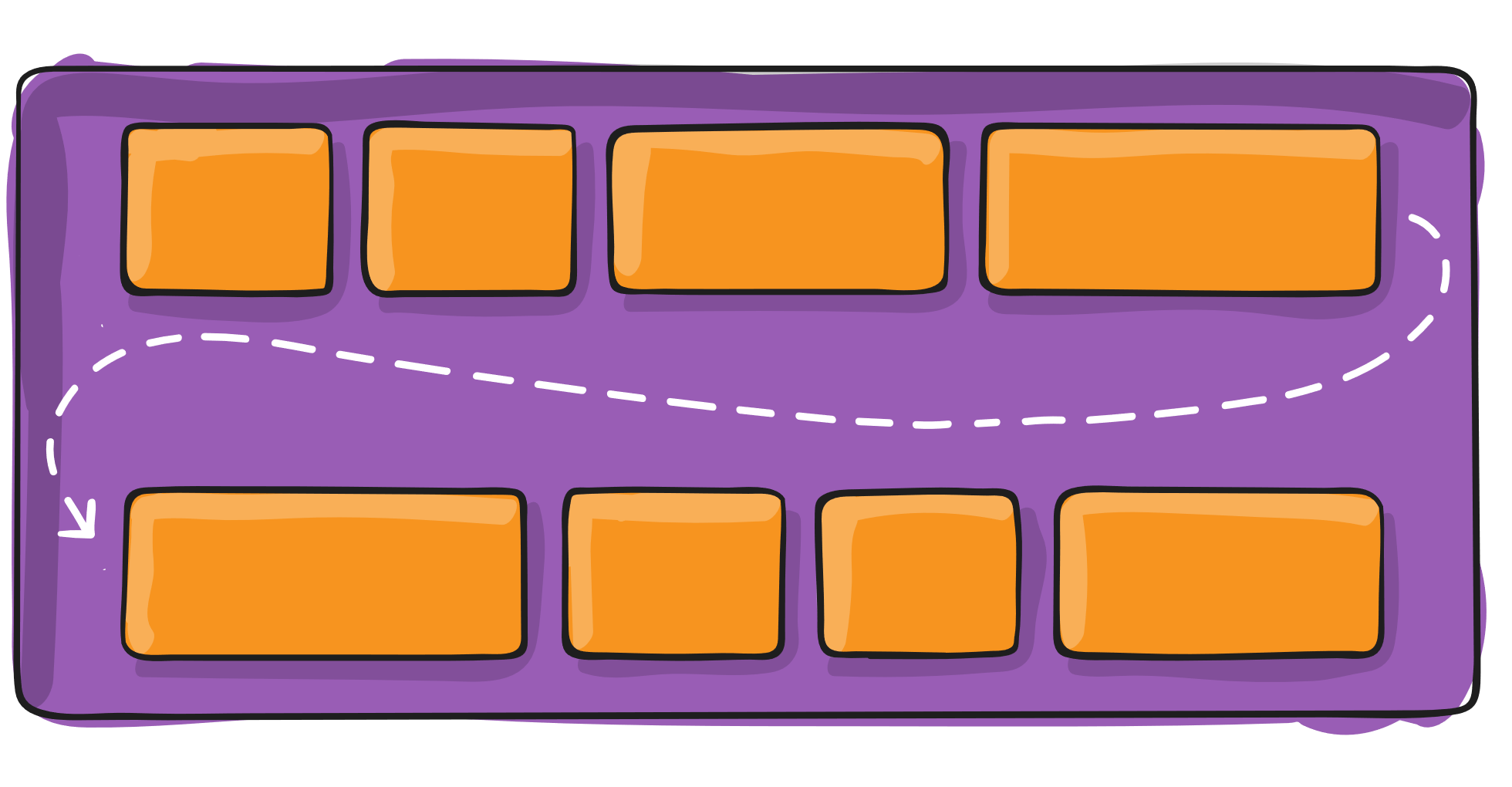


.container {

flex-direction: row | row-reverse | column | column-reverse;}

* row (default): left to right in ltr; right to left in rtl
* row-reverse: right to left in ltr; left to right in rtl
* column: same as row but top to bottom
* column-reverse: same as row-reverse but bottom to top

#### flex-wrap



By default, flex items will all try to fit onto one line. You can and allow the items to wrap as needed

.container {

flex-wrap: nowrap | wrap | wrap-reverse;

}

* nowrap (default): all flex items will be on one line
* wrap: flex items will wrap onto multiple lines, from top to bottom.
* wrap-reverse: flex items will wrap onto multiple lines from bottom to top.

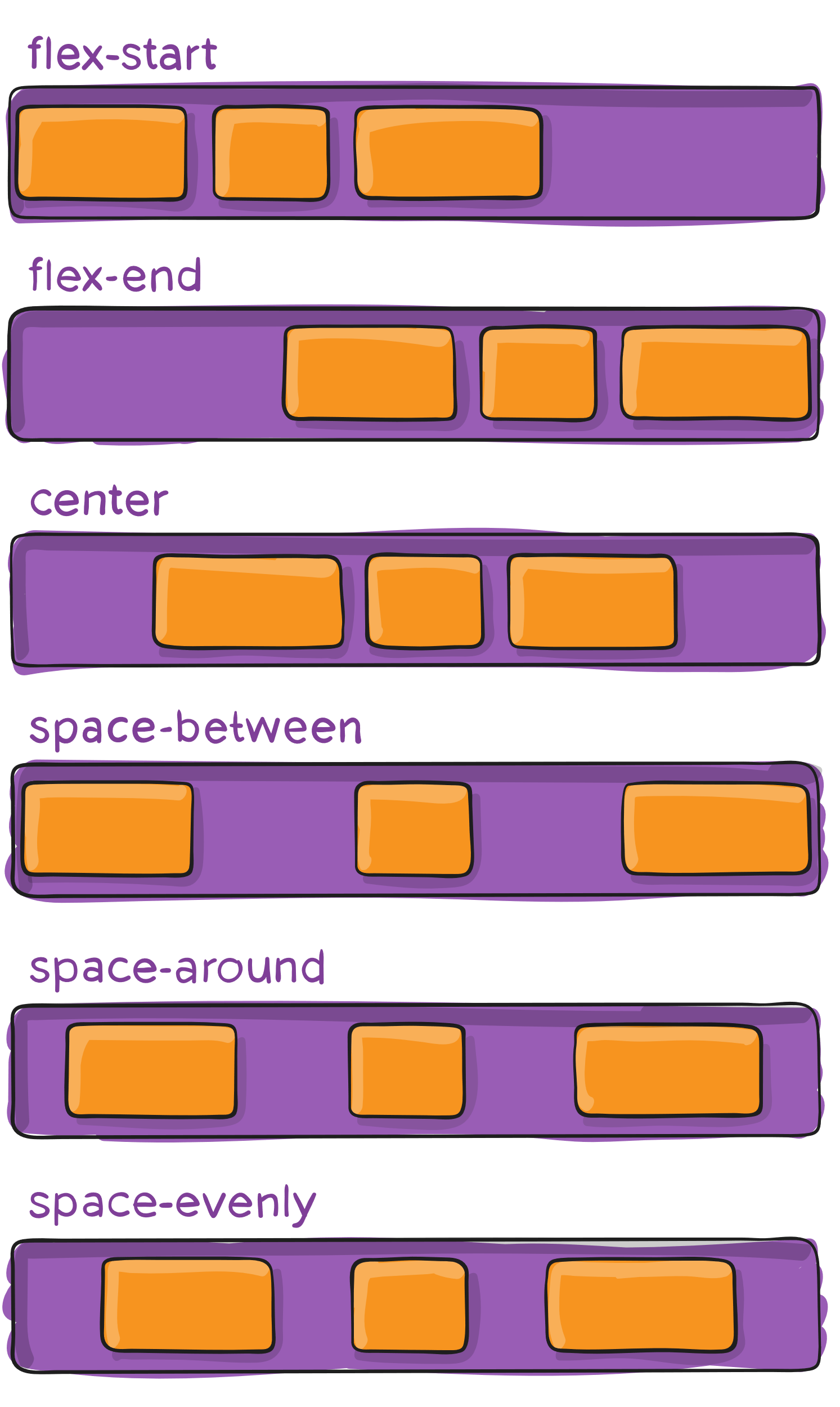
#### flex-flow

This is a shorthand for the flex-direction and flex-wrap properties, which together define the flex container’s main and cross axes. The default value is row nowrap.

.container {

flex-flow: column wrap;}

#### justify-content



This defines the alignment along the main axis. It helps distribute extra free space leftover when either, all the flex items on a line are inflexible, or are flexible but have reached their maximum size. It also exerts some control over the alignment of items when they overflow the line.

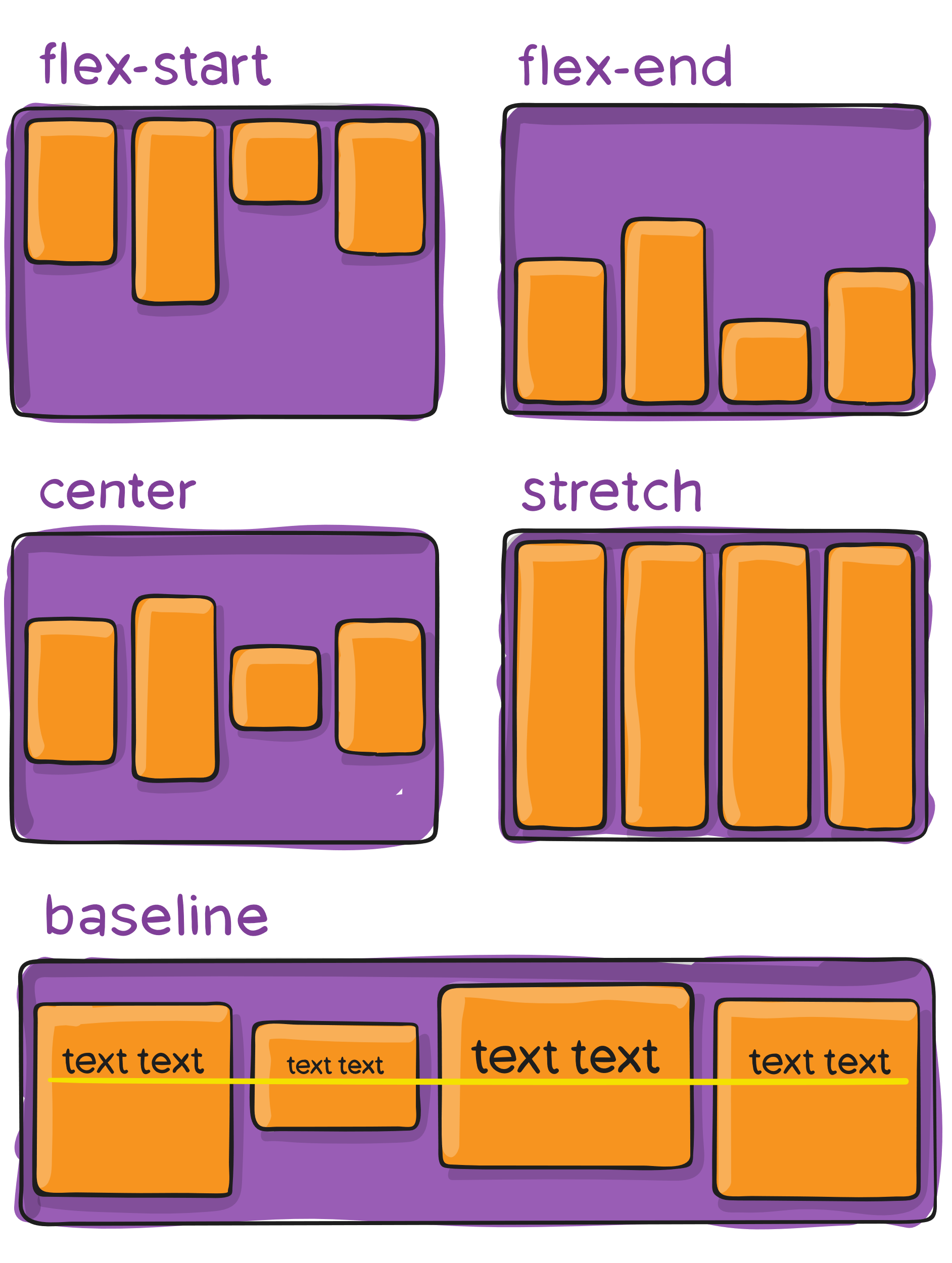
.container {

justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly | start | end | left | right ... + safe | unsafe;

}

* flex-start (default): items are packed toward the start of the flex-direction.
* flex-end: items are packed toward the end of the flex-direction.
* start: items are packed toward the start of the writing-mode direction.
* end: items are packed toward the end of the writing-mode direction.
* left: items are packed toward left edge of the container, unless that doesn’t make sense with the flex-direction, then it behaves like start.
* right: items are packed toward right edge of the container, unless that doesn’t make sense with the flex-direction, then it behaves like end.
* center: items are centered along the line
* space-between: items are evenly distributed in the line; first item is on the start line, last item on the end line
* space-around: items are evenly distributed in the line with equal space around them. Note that visually the spaces aren’t equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies.
* space-evenly: items are distributed so that the spacing between any two items (and the space to the edges) is equal.

#### align-items



This defines the default behavior for how flex items are laid out along the **cross axis** on the current line. Think of it as the justify-content version for the cross-axis (perpendicular to the main-axis).

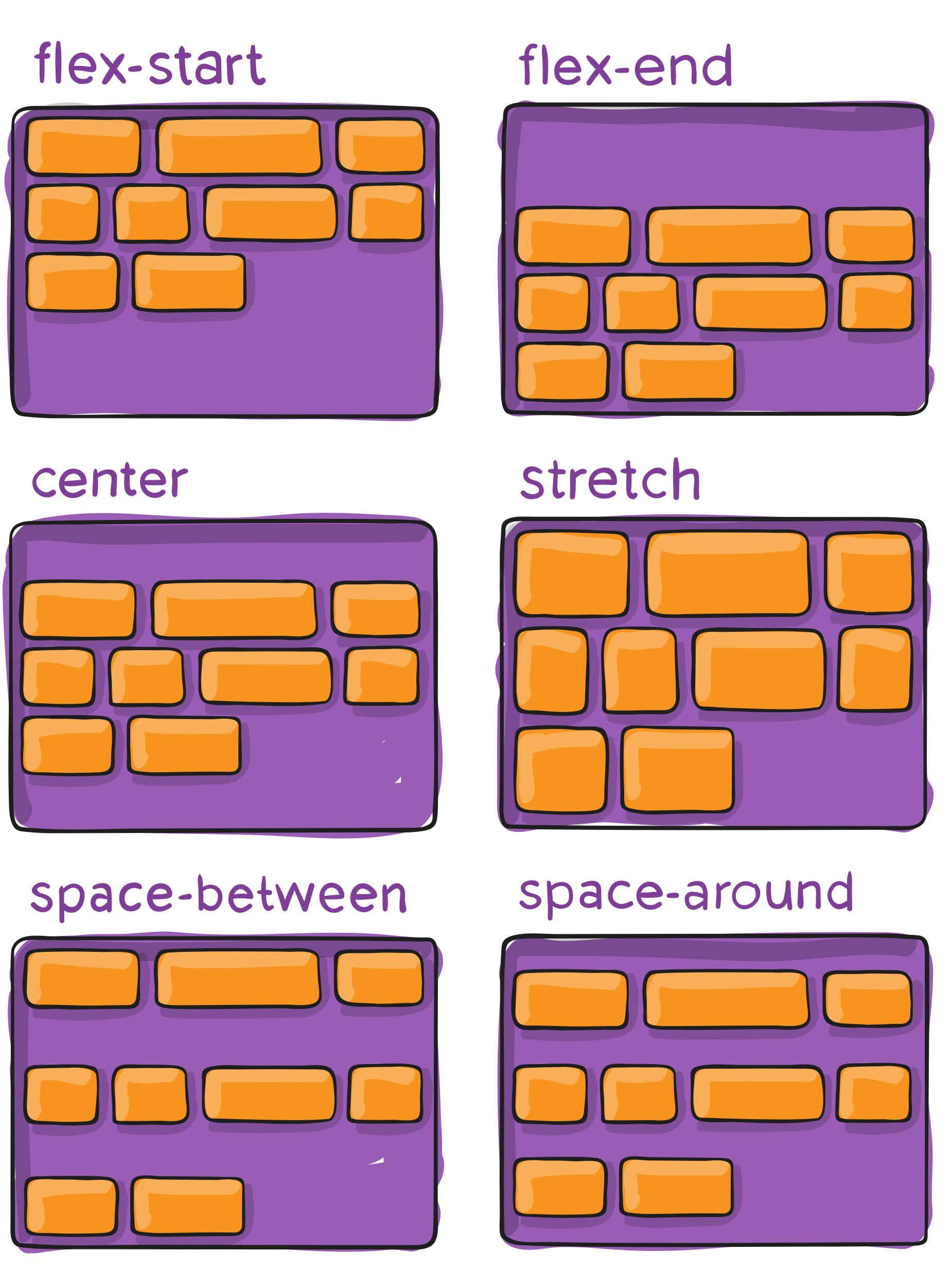
.container {

align-items: stretch | flex-start | flex-end | center | baseline | first baseline | last baseline | start | end | self-start | self-end + ... safe | unsafe;

}

* stretch (default): stretch to fill the container (still respect min-width/max-width)
* flex-start / start / self-start: items are placed at the start of the cross axis. The difference between these is subtle, and is about respecting the flex-direction rules or the writing-mode rules.
* flex-end / end / self-end: items are placed at the end of the cross axis. The difference again is subtle and is about respecting flex-direction rules vs. writing-mode rules.
* center: items are centered in the cross-axis
* baseline: items are aligned such as their baselines align

#### align-content



This aligns a flex container’s lines within when there is extra space in the cross-axis, similar to how justify-content aligns individual items within the main-axis.

**Note:** This property only takes effect on multi-line flexible containers, where flex-wrap is set to either wrap or wrap-reverse). A single-line flexible container (i.e. where flex-wrap is set to its default value, no-wrap) will not reflect align-content.

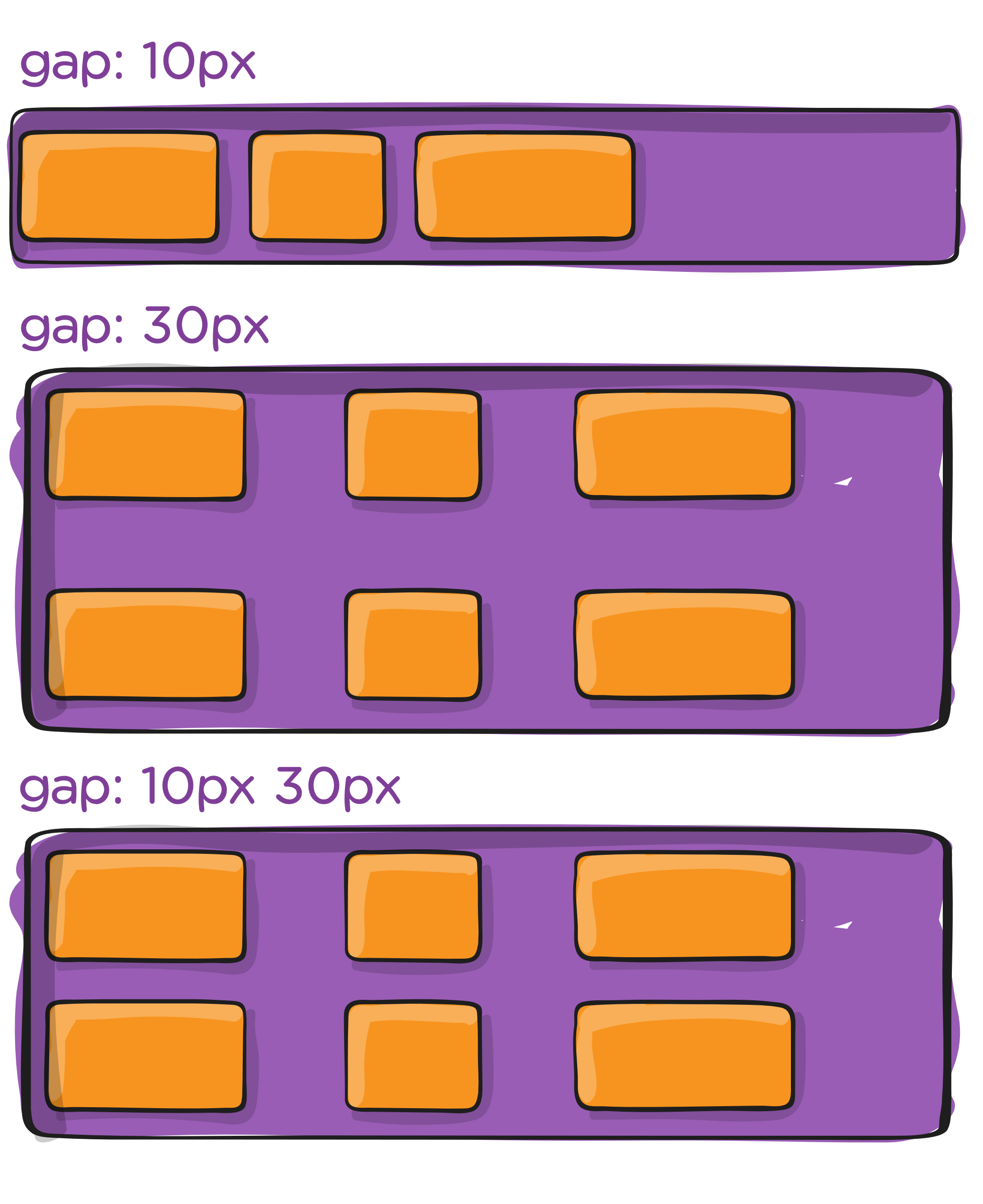
.container {

align-content: flex-start | flex-end | center | space-between | space-around | space-evenly | stretch | start | end | baseline | first baseline | last baseline + ... safe | unsafe;

}

* normal (default): items are packed in their default position as if no value was set.
* flex-start / start: items packed to the start of the container. The (more supported) flex-start honors the flex-direction while start honors the writing-mode direction.
* flex-end / end: items packed to the end of the container. The (more support) flex-end honors the flex-direction while end honors the writing-mode direction.
* center: items centered in the container
* space-between: items evenly distributed; the first line is at the start of the container while the last one is at the end
* space-around: items evenly distributed with equal space around each line
* space-evenly: items are evenly distributed with equal space around them
* stretch: lines stretch to take up the remaining space

#### gap, row-gap, column-gap



[The gap property](https://css-tricks.com/almanac/properties/g/gap/) explicitly controls the space between flex items. It applies that spacing only between items not on the outer edges.

.container {

display: flex;

...

gap: 10px;

gap: 10px 20px; /\* row-gap column gap \*/

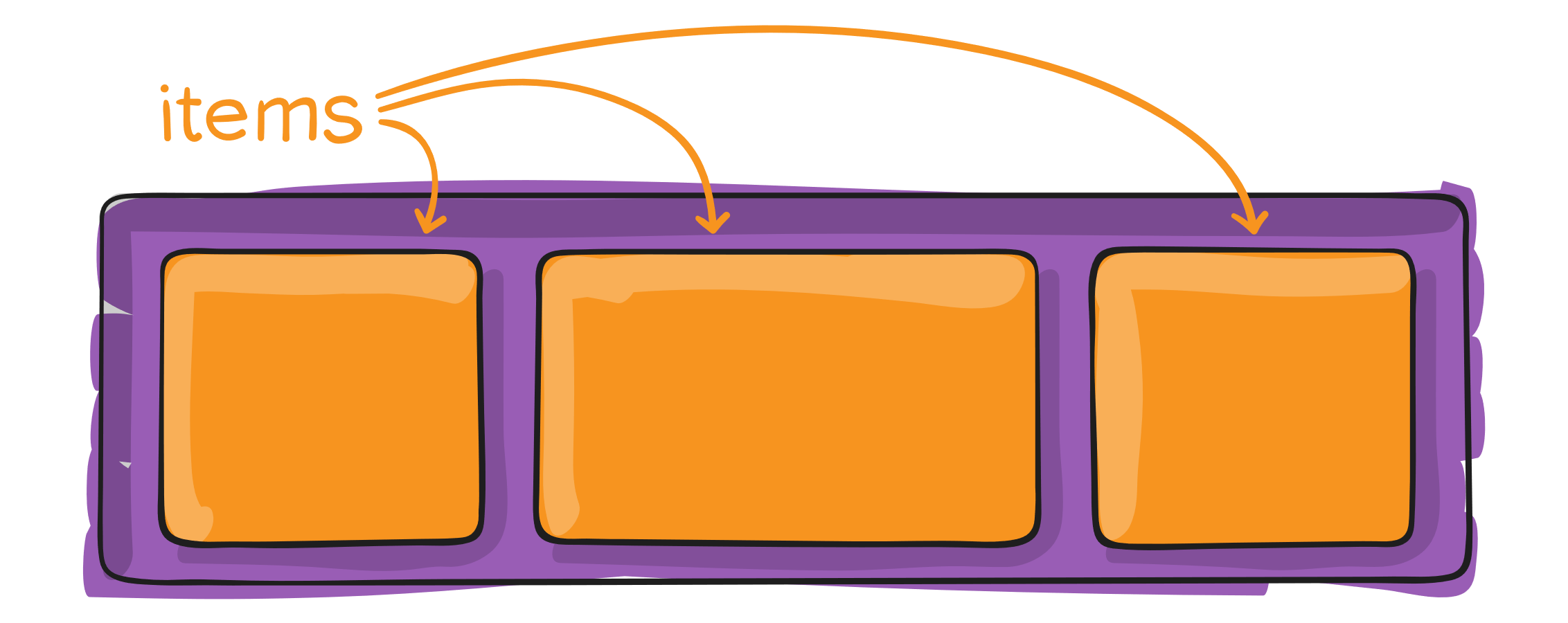
row-gap: 10px;

column-gap: 20px;

}

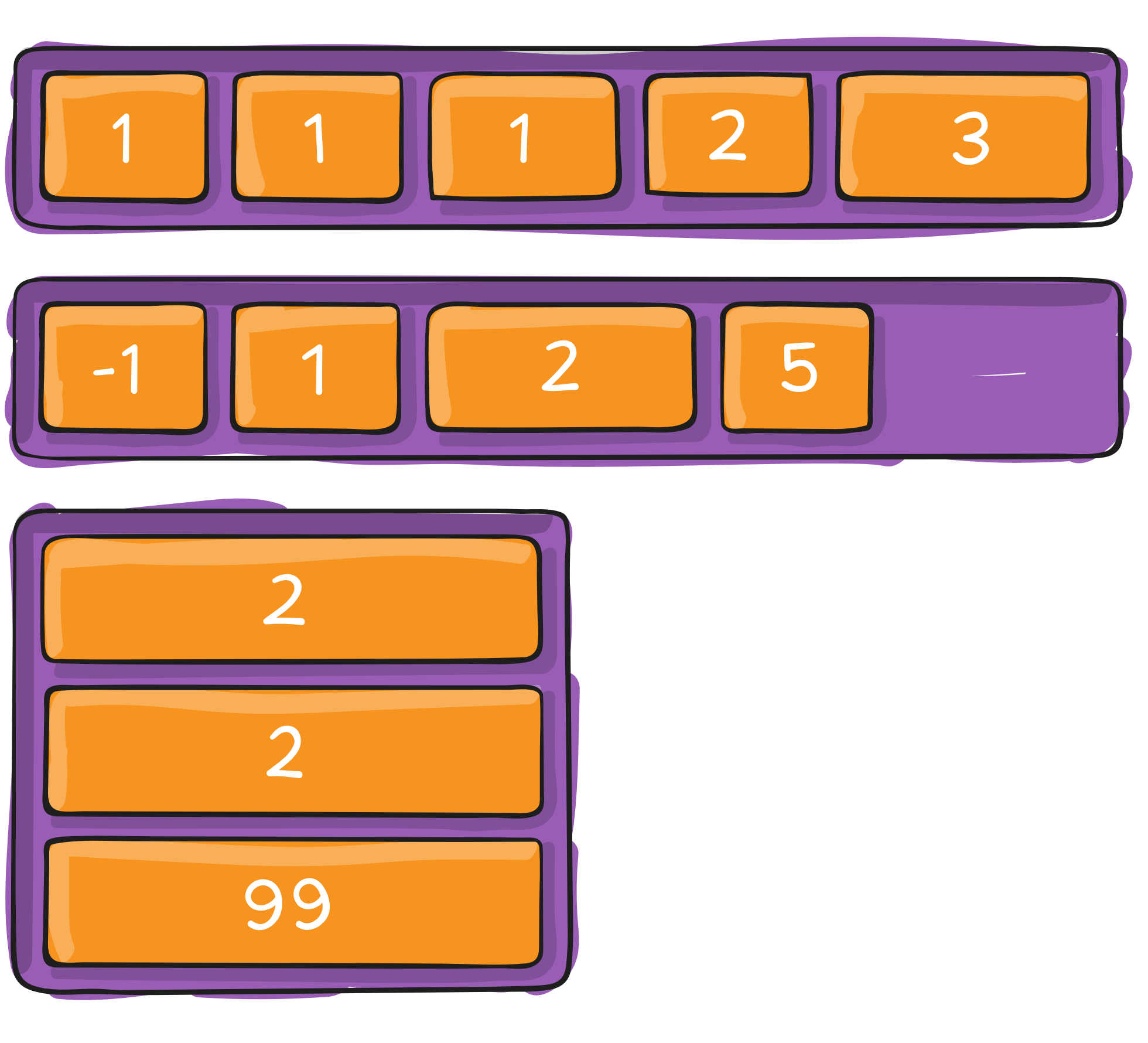
The behavior could be thought of as a minimum gutter, as if the gutter is bigger somehow (because of something like justify-content: space-between;) then the gap will only take effect if that space would end up smaller.

It is not exclusively for flexbox, gap works in grid and multi-column layout as well.



## Properties for the Children (flex items)

#### order



By default, flex items are laid out in the source order. However, the order property controls the order in which they appear in the flex container.

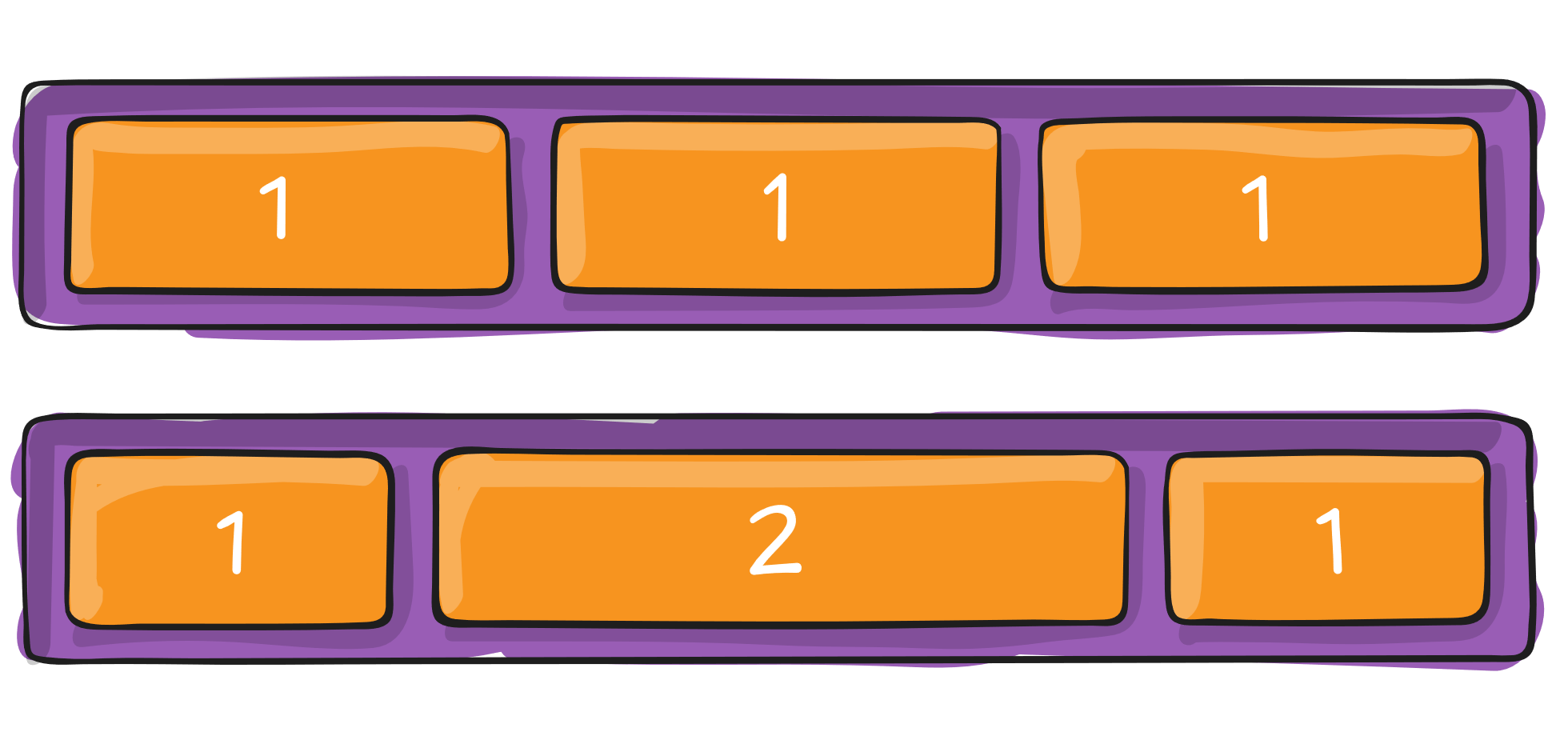
.item {

order: 5; /\* default is 0 \*/

}

Items with the same order revert to source order.

#### flex-grow



This defines the ability for a flex item to grow if necessary. It accepts a unitless value that serves as a proportion. It dictates what amount of the available space inside the flex container the item should take up.

If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all children. If one of the children has a value of 2, that child would take up twice as much of the space either one of the others (or it will try, at least).

.item {

flex-grow: 4; /\* default 0 \*/

}

Negative numbers are invalid.

#### flex-shrink

This defines the ability for a flex item to shrink if necessary.

.item {

flex-shrink: 3; /\* default 1 \*/

}

Negative numbers are invalid.

#### flex-basis

This defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword. The auto keyword means “look at my width or height property” (which was temporarily done by the main-size keyword until deprecated). The content keyword means “size it based on the item’s content” – this keyword isn’t well supported yet, so it’s hard to test and harder to know what its brethren max-content, min-content, and fit-content do.

.item {

flex-basis: | auto; /\* default auto \*/

}

If set to 0, the extra space around content isn’t factored in. If set to auto, the extra space is distributed based on its flex-grow value. [See this graphic.](http://www.w3.org/TR/css3-flexbox/images/rel-vs-abs-flex.svg)

#### flex

This is the shorthand for flex-grow, flex-shrink and flex-basis combined. The second and third parameters (flex-shrink and flex-basis) are optional. The default is 0 1 auto, but if you set it with a single number value, like flex: 5;, that changes the flex-basis to 0%, so it’s like setting flex-grow: 5; flex-shrink: 1; flex-basis: 0%;.

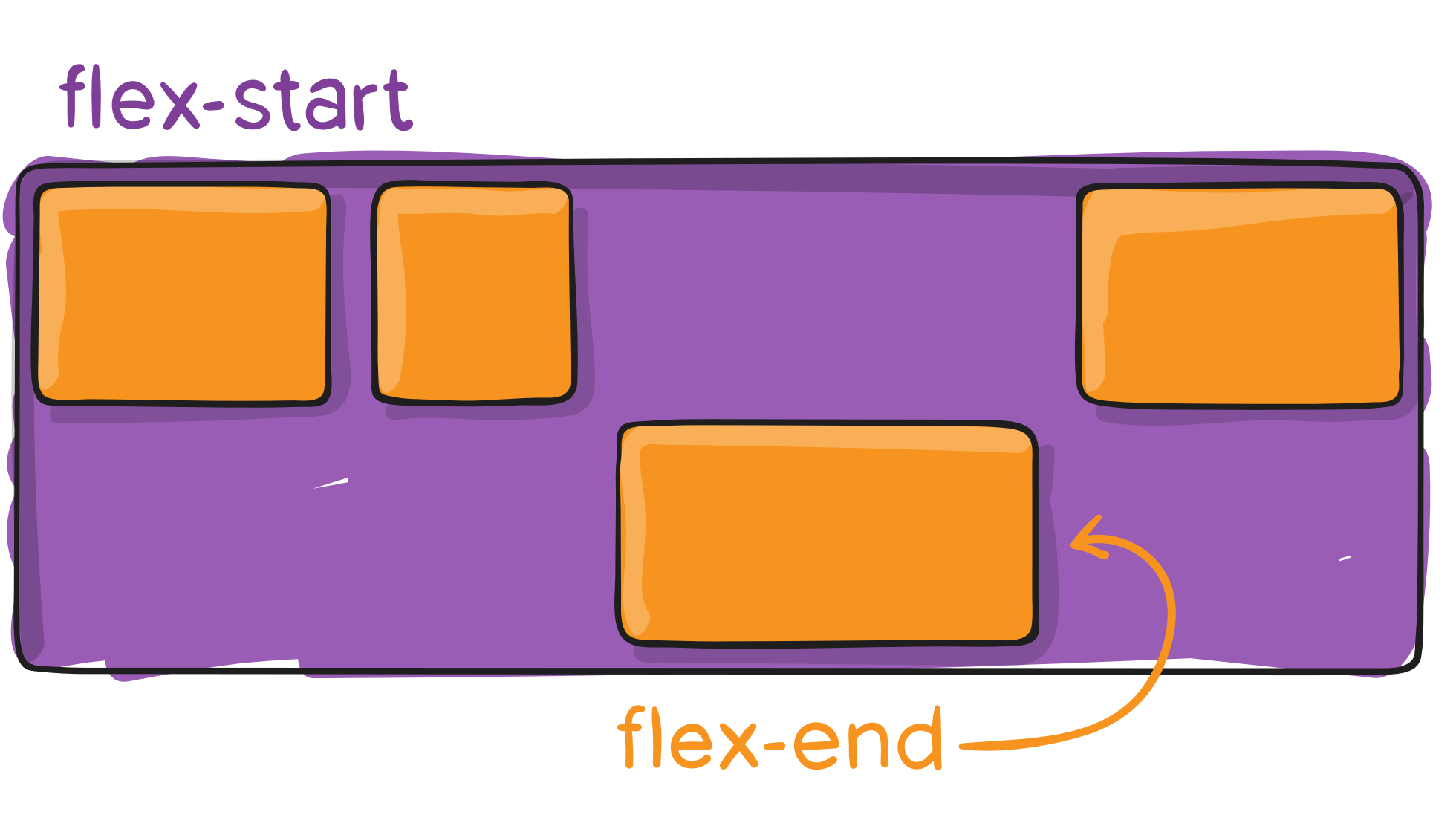
.item {

flex: none | [ <'flex-grow'> <'flex-shrink'>? || <'flex-basis'> ]

}

**It is recommended that you use this shorthand property** rather than set the individual properties. The shorthand sets the other values intelligently.

#### align-self



This allows the default alignment (or the one specified by align-items) to be overridden for individual flex items.

Please see the align-items explanation to understand the available values.

.item {

align-self: auto | flex-start | flex-end | center | baseline | stretch;

}

Note that float, clear and vertical-align have no effect on a flex item./