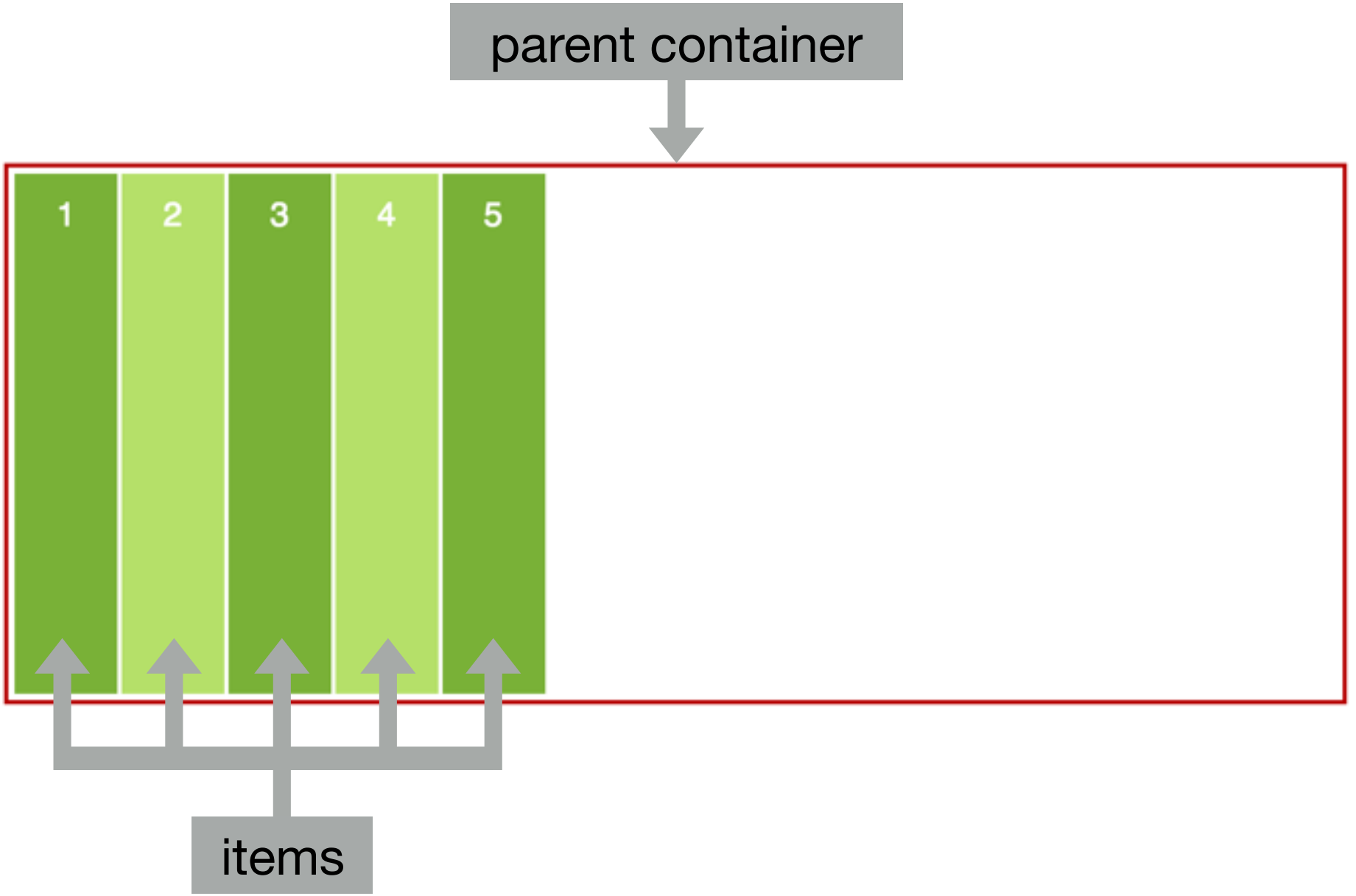
CSS3

FLEXBOX

The **Flexible Box module** (or flexbox) provides us with more a efficient way to lay out, align and distribute space among items in a container.

Flexbox focusses on **two components** - the parent container and the child items.


parent container
1
2
3
4
5
items

Flexbox allows containers to alter the **width, height and order** of their child items so that these items can fill the available space.

Properties for the parent

display: flex

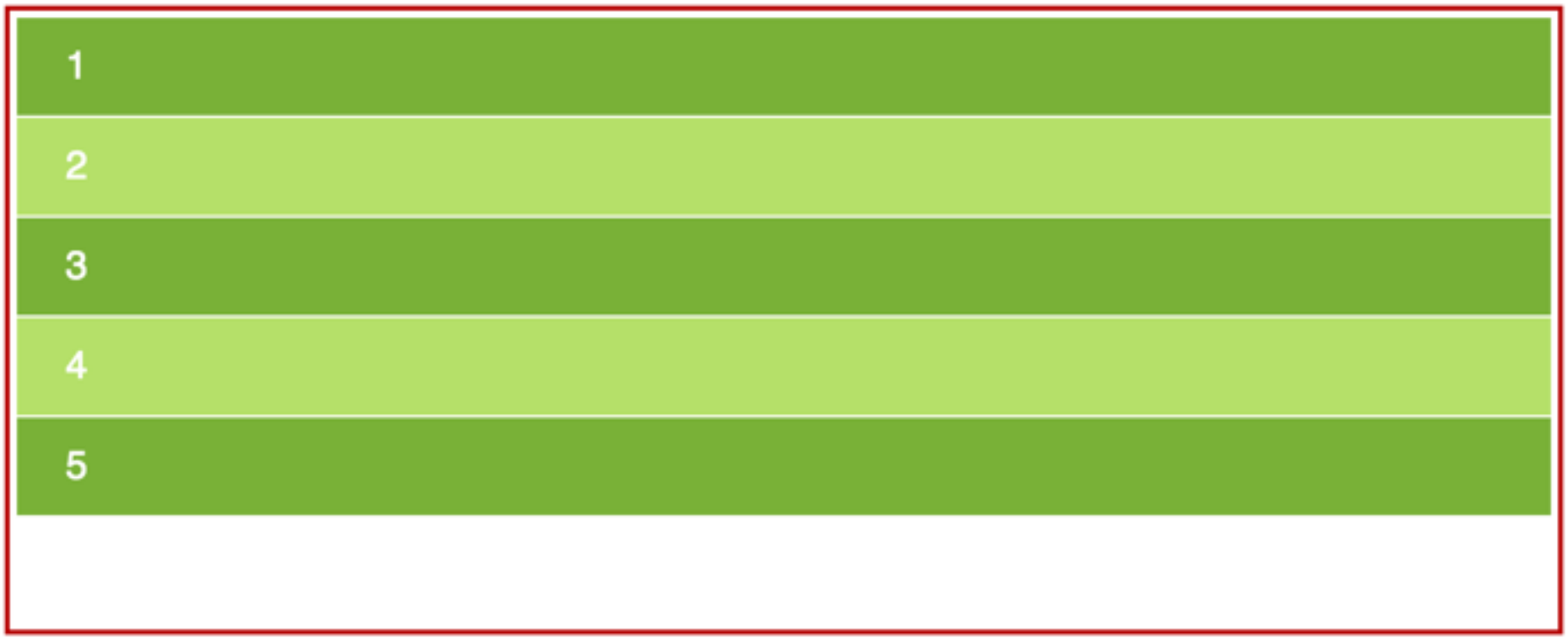
The core of flexbox are the **display: flex** and **display: inline-flex** values.

These two values **define whether an element is a flex container**; block or inline. The values enable a flex context for all of the parent element's direct children.

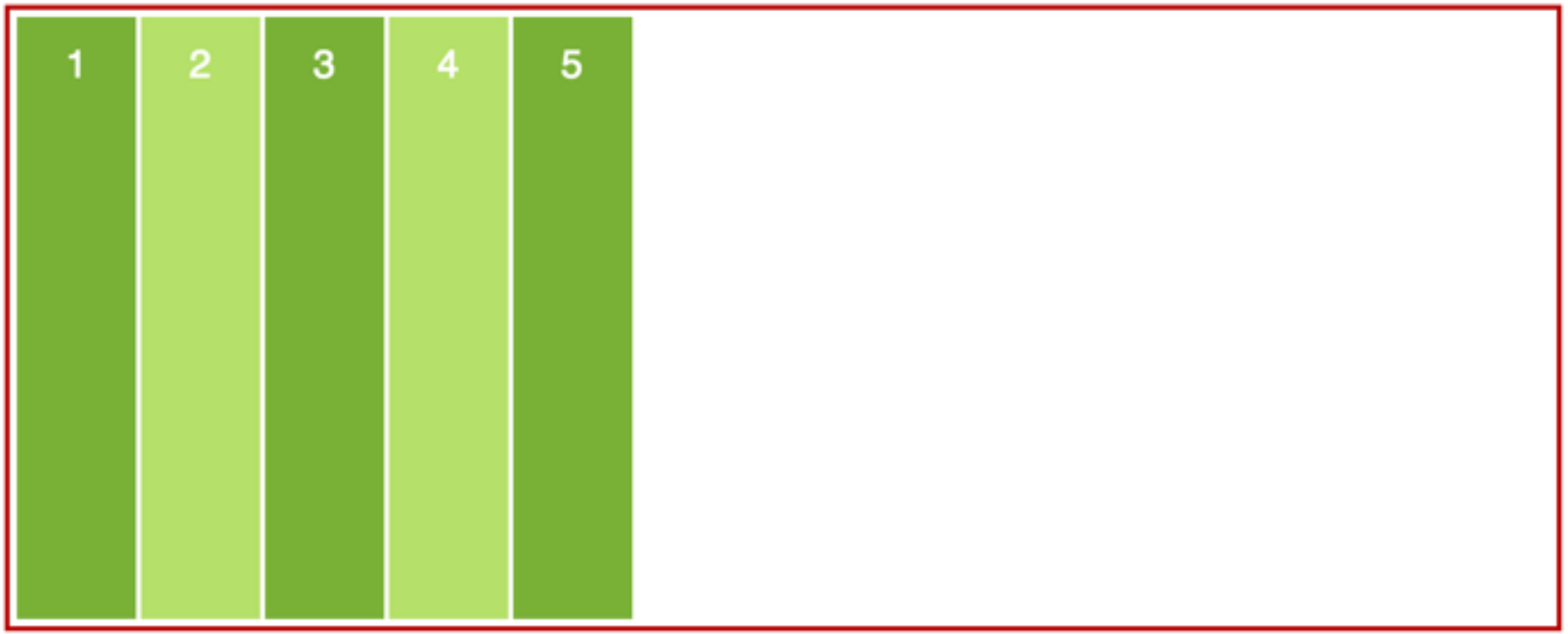
```
.parent
{
    display: flex;
}

.parent
{
    display: inline-flex;
}
```

As soon as these properties are applied, child items will **change from block-level items** into columns.

Before display: flex is applied



After display: flex is applied



flex-direction

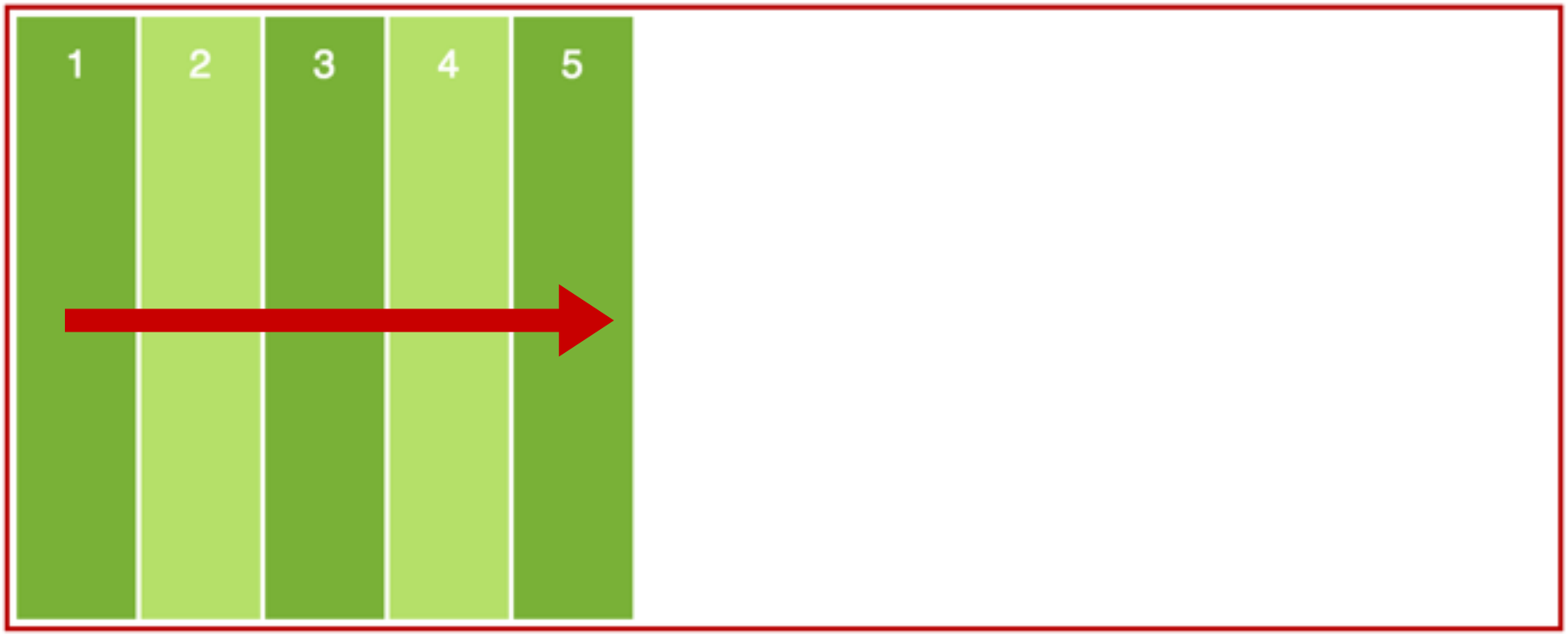
The **flex-direction property** specifies the direction of the flexible items within a parent.

```
.parent { flex-direction: ; }
```

The **row value** sets flexible items to be displayed horizontally, as a row. This is the default value.

```
.parent { flex-direction: row; }
```

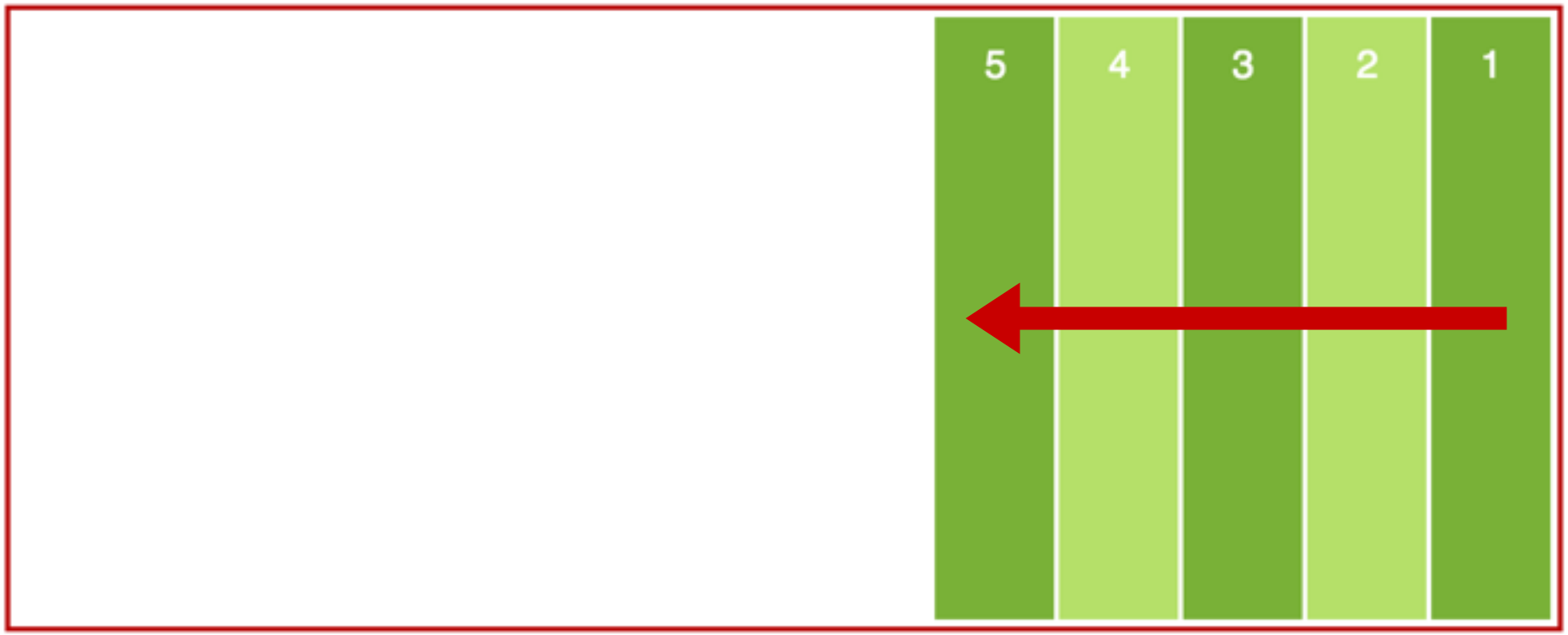
flex-direction: row



The **row-reverse value** is the same as row, but in reverse order.

```
.parent { flex-direction: row-reverse; }
```

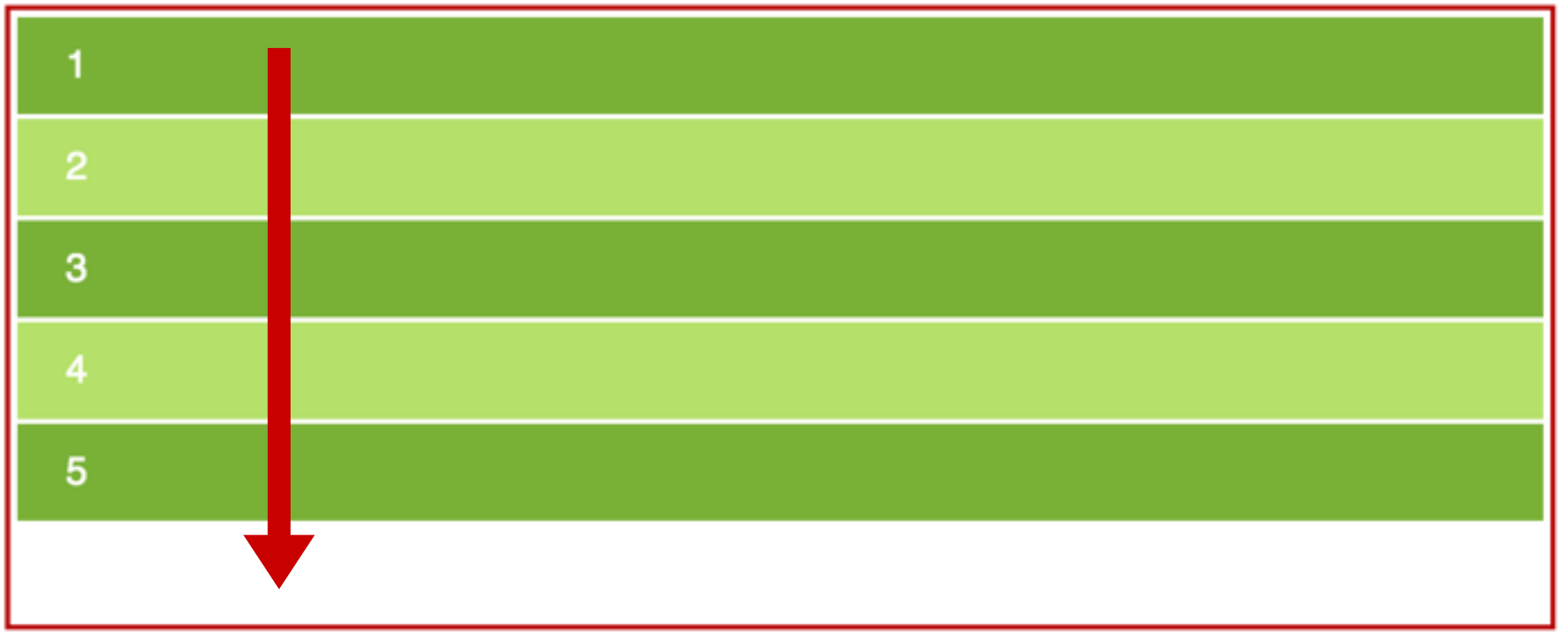
flex-direction: row-reverse



The **column value** sets the flexible items to be displayed vertically, as a column.

```
.parent { flex-direction: column; }
```

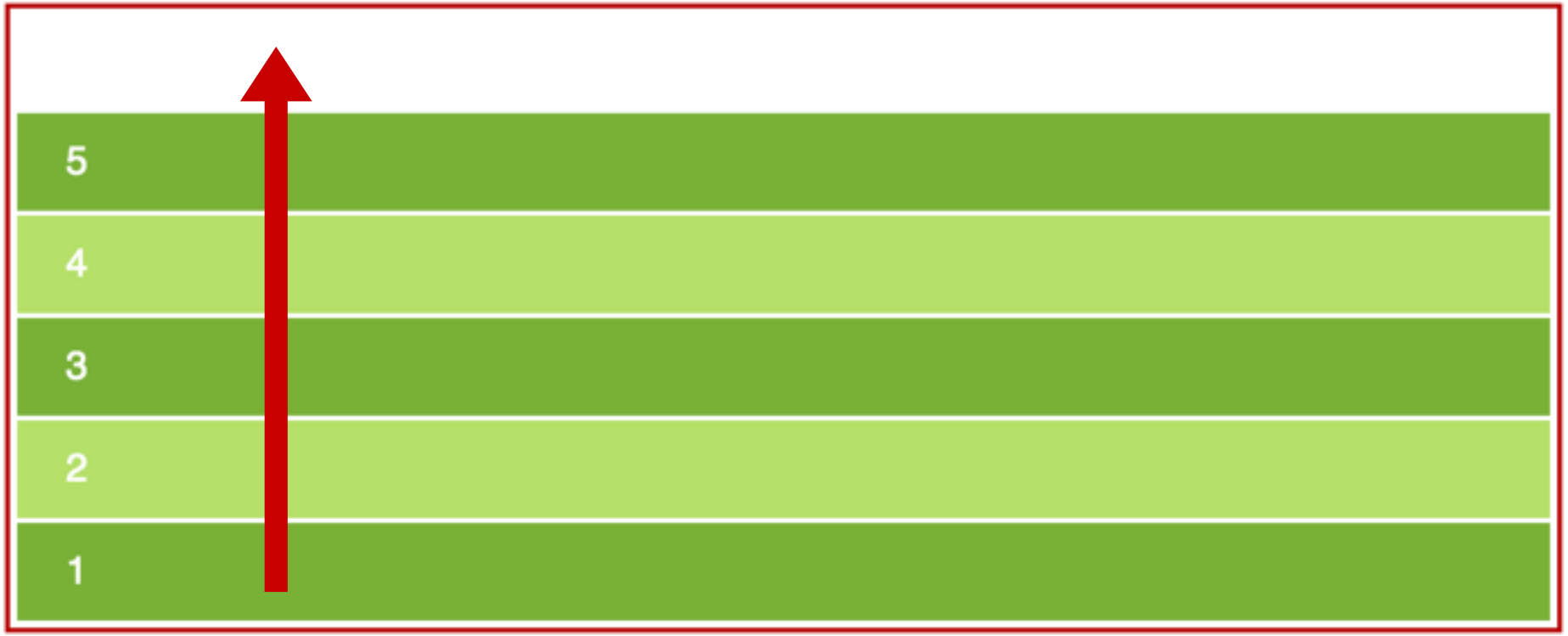
flex-direction: column



The **column-reverse value** is the same as row, but in reverse order.

```
.parent { flex-direction: column-reverse; }
```

flex-direction: column-reverse



flex-wrap

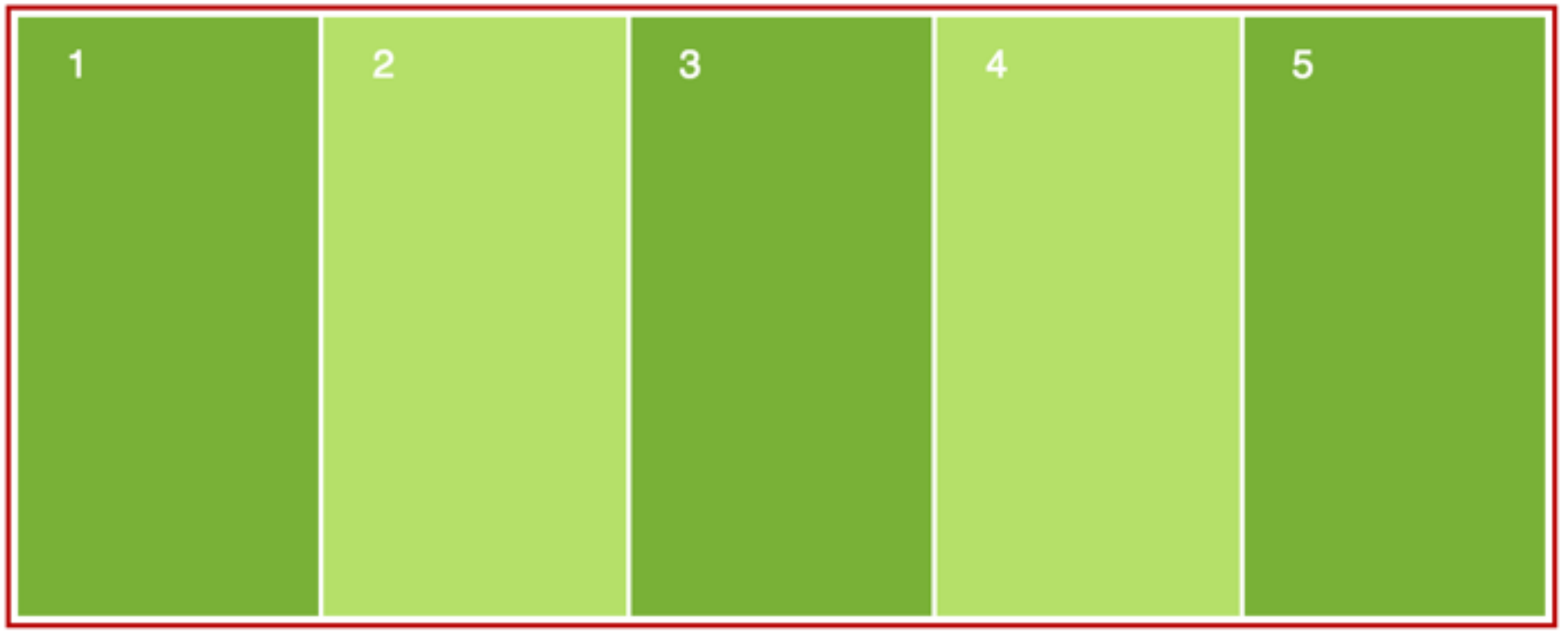
The **flex-wrap property** specifies whether the flexible items should wrap (move to a new line) or not.

```
.parent { flex-wrap: ; }
```

The **nowrap value** specifies that the flexible items will not wrap. This is the default value.

```
.parent { flex-wrap: nowrap; }
```

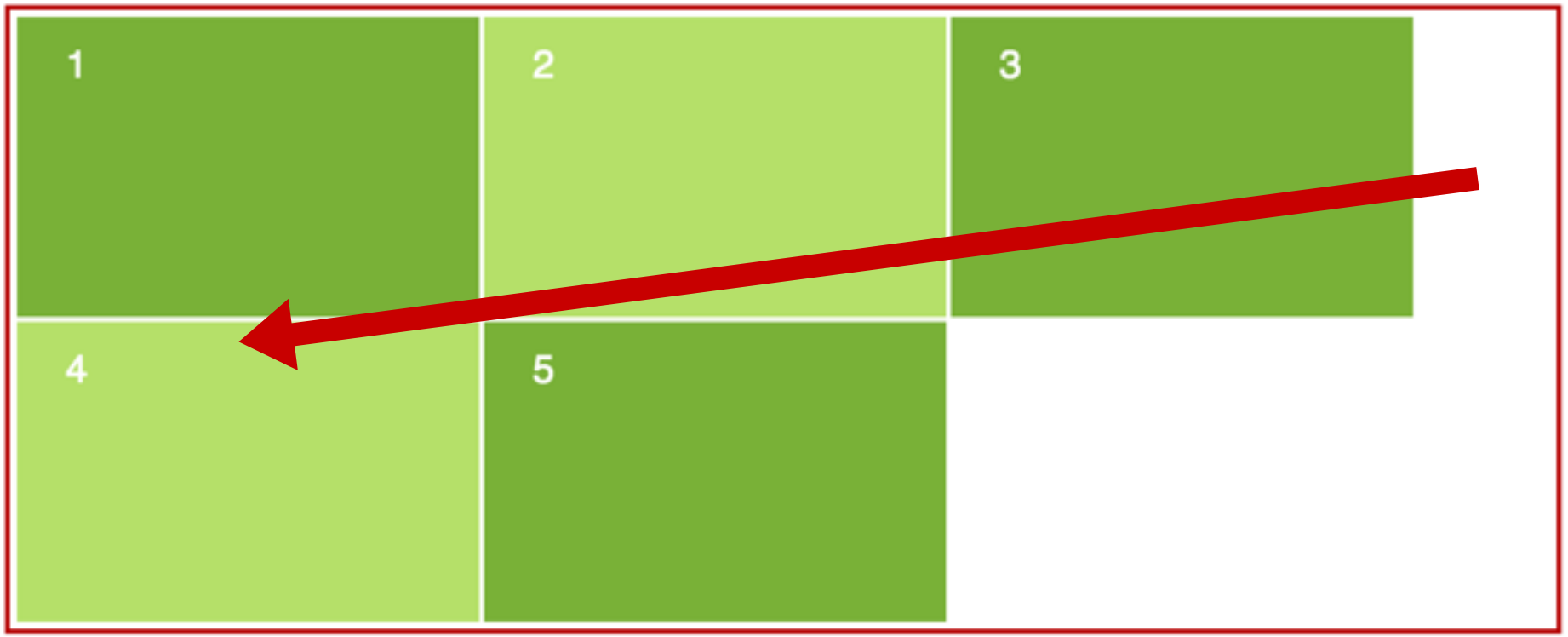
flex-wrap: nowrap



The **wrap value** specifies that the flexible items will wrap if necessary.

```
.parent { flex-wrap: wrap; }
```

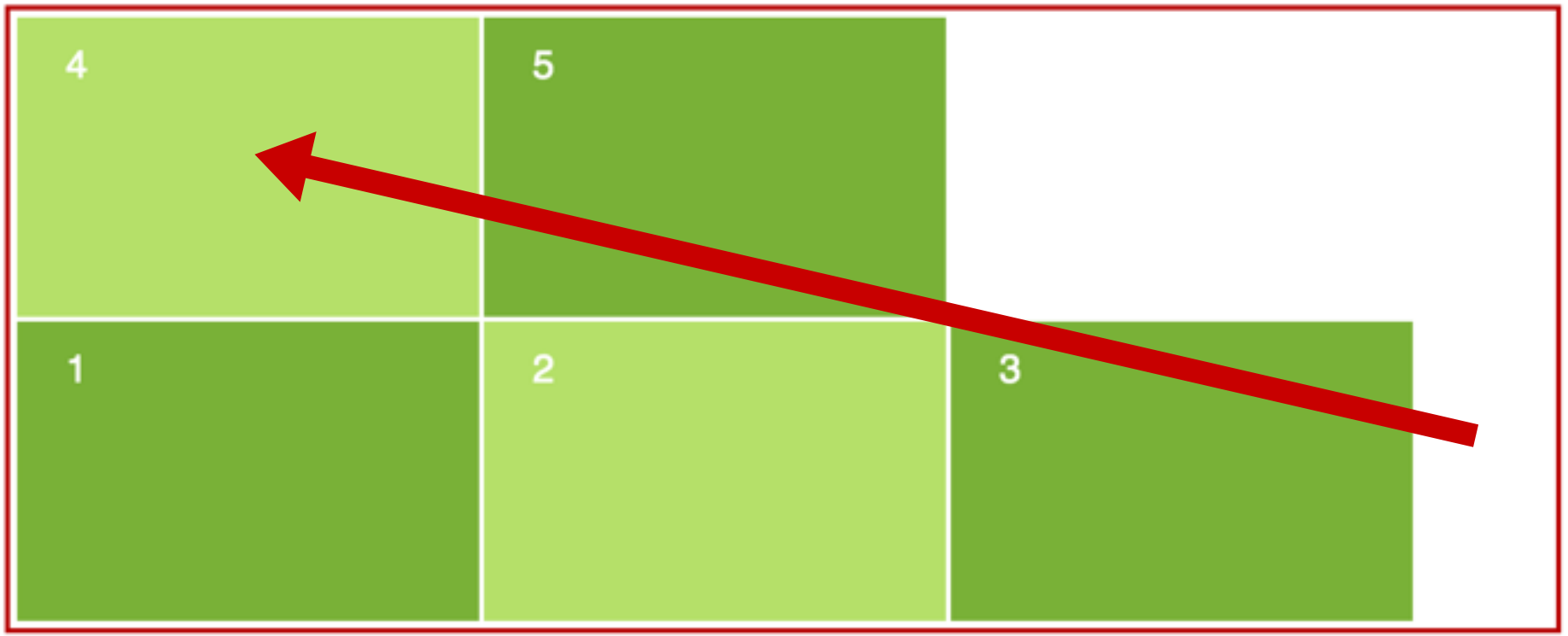
flex-wrap: wrap



The **wrap-reverse value** specifies that the flexible items will wrap, if necessary, in reverse order.

```
.parent { flex-wrap: wrap-reverse; }
```

flex-wrap: wrap-reverse



justify-content

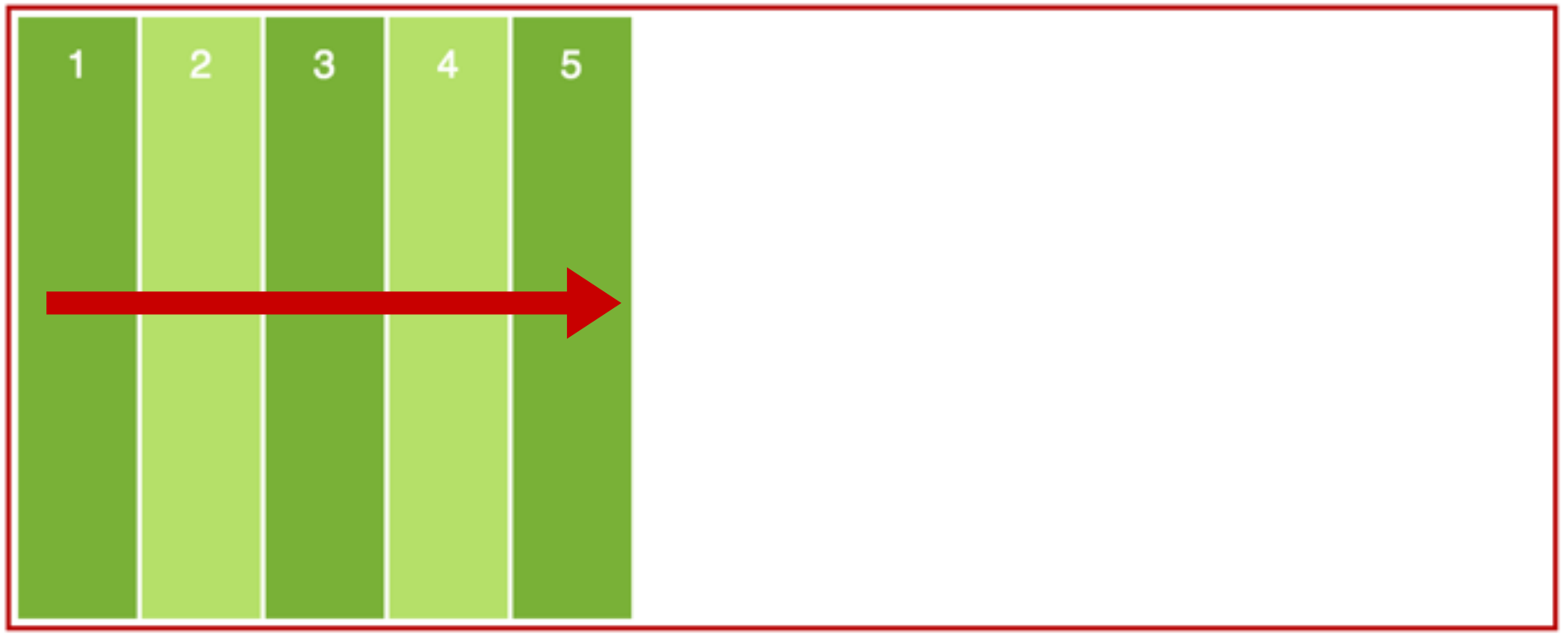
The **justify-content property** aligns the flexible container's items when the items do not use all available space on the main-axis (horizontally).

```
.parent { justify-content: ; }
```

The **flex-start value** sets items to be positioned at the beginning of the container. This is the default value.

```
.parent { justify-content: flex-start; }
```

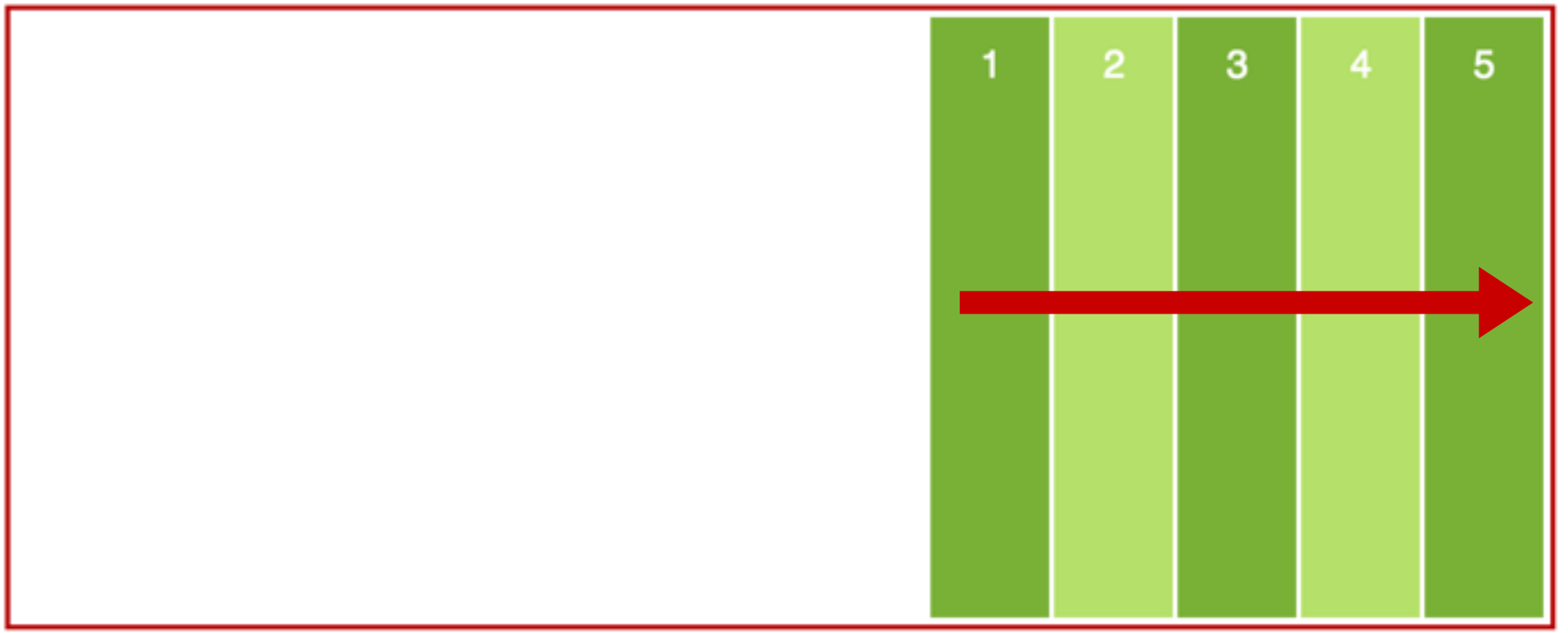
justify-content: flex-start



The **flex-end value** sets items to be positioned at the end of the container.

```
.parent { justify-content: flex-end; }
```

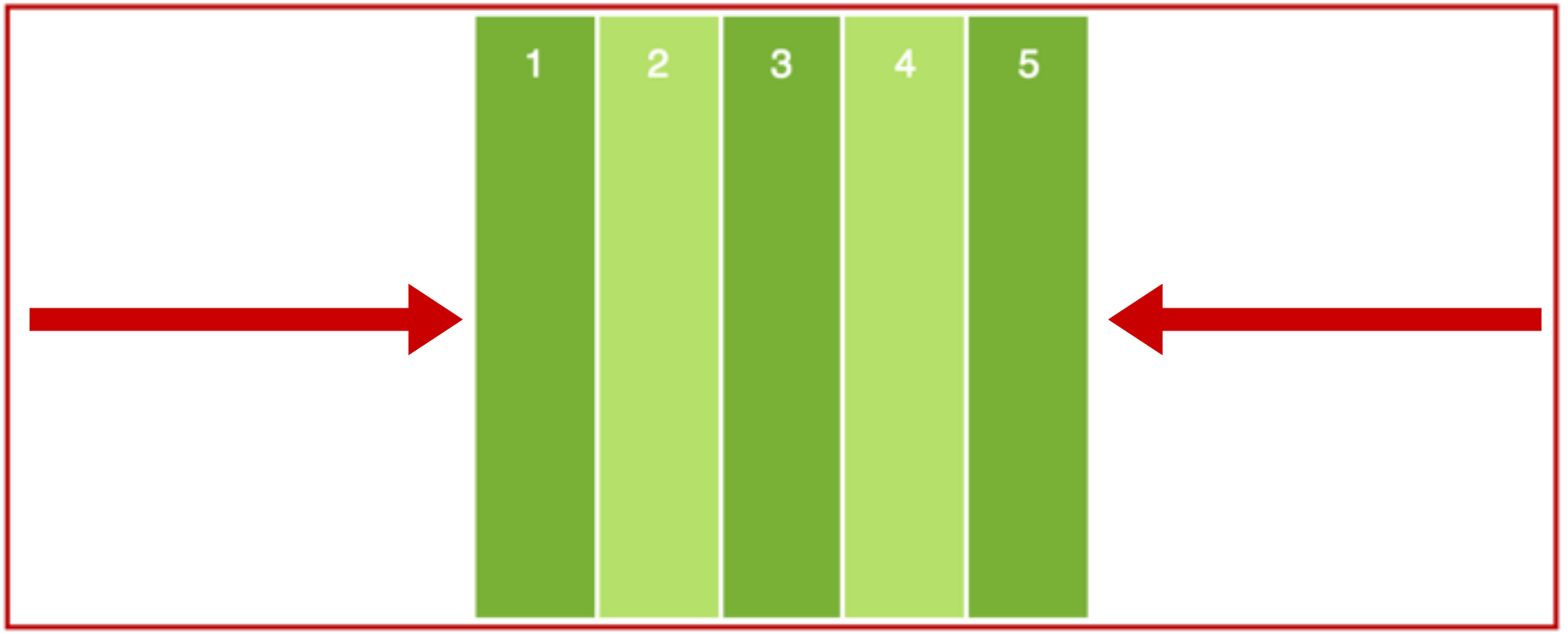
justify-content: flex-end



The **center value** sets items to be positioned at the center of the container.

```
.parent { justify-content: center; }
```

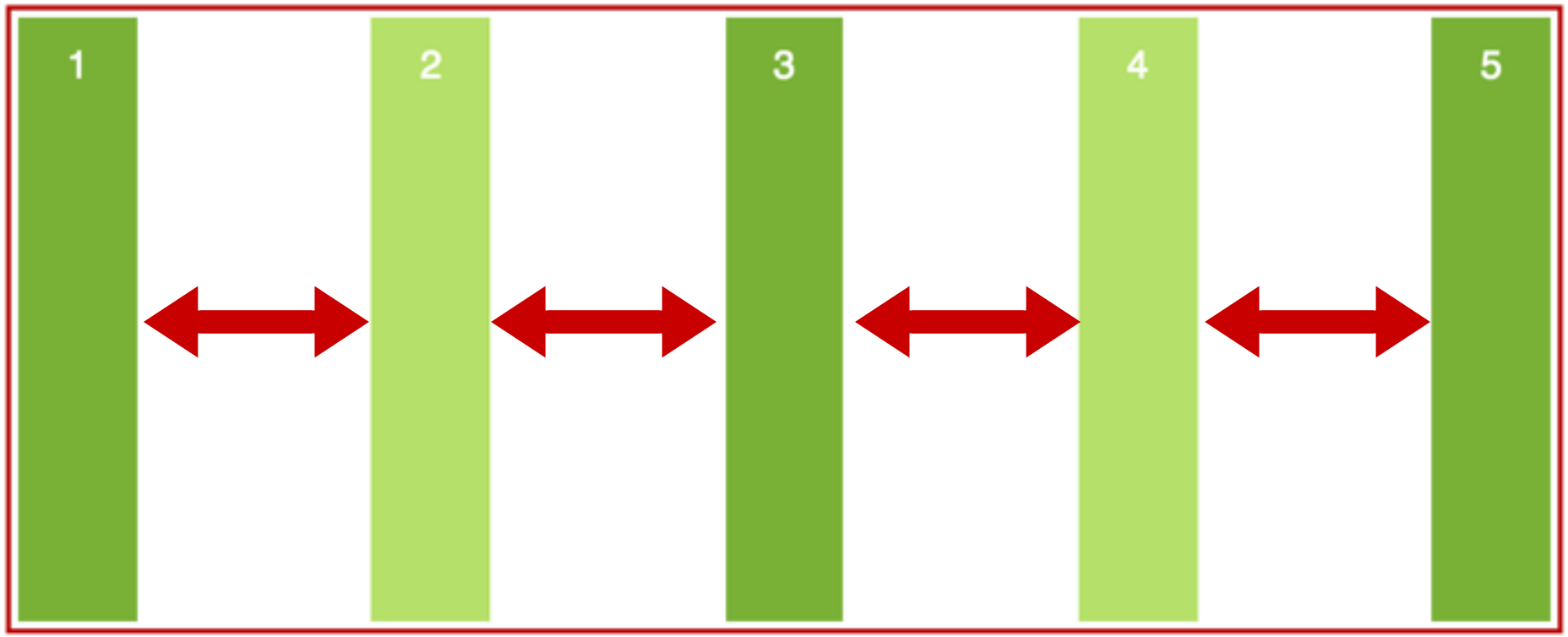
justify-content: center



The **space-between value** sets items to be positioned with space between the lines.

```
.parent { justify-content: space-between; }
```

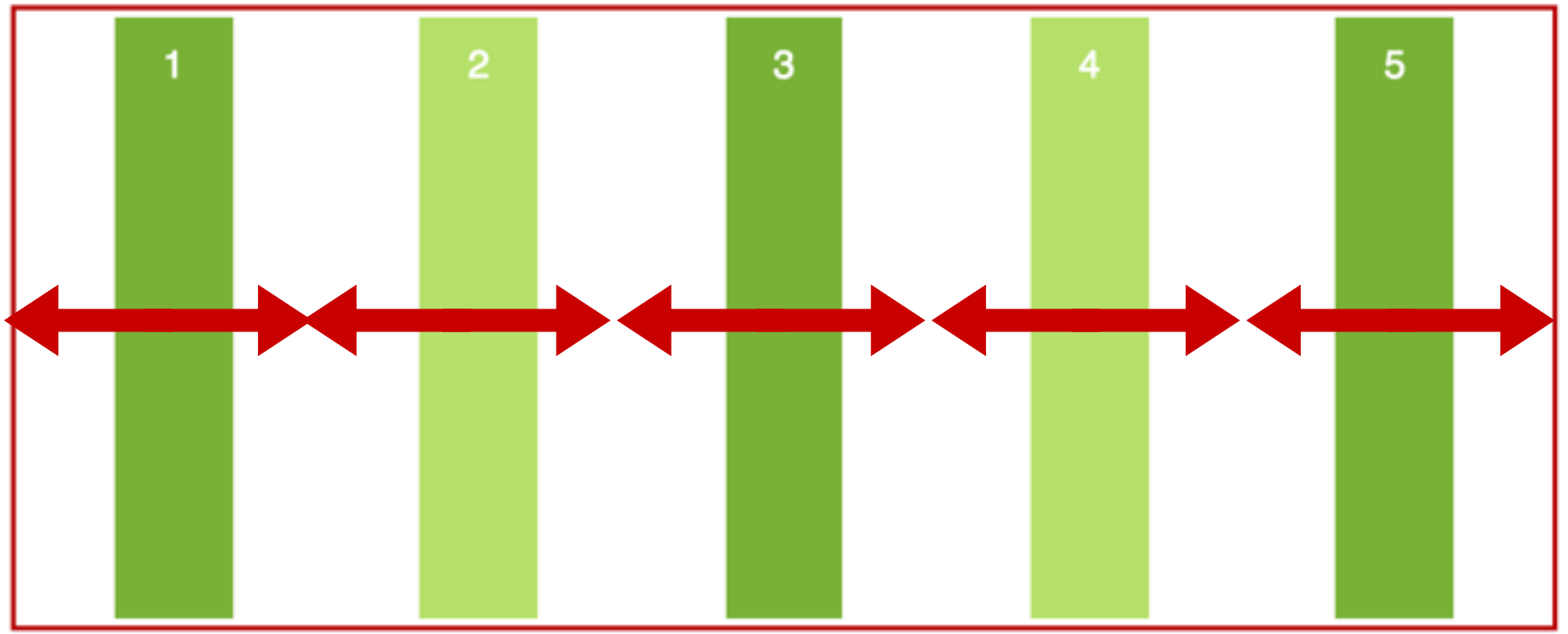
justify-content: space-between



The **space-around value** sets items to be positioned with space before, between, and after the lines.

```
.parent { justify-content: space-around; }
```

justify-content: space-around



align-items

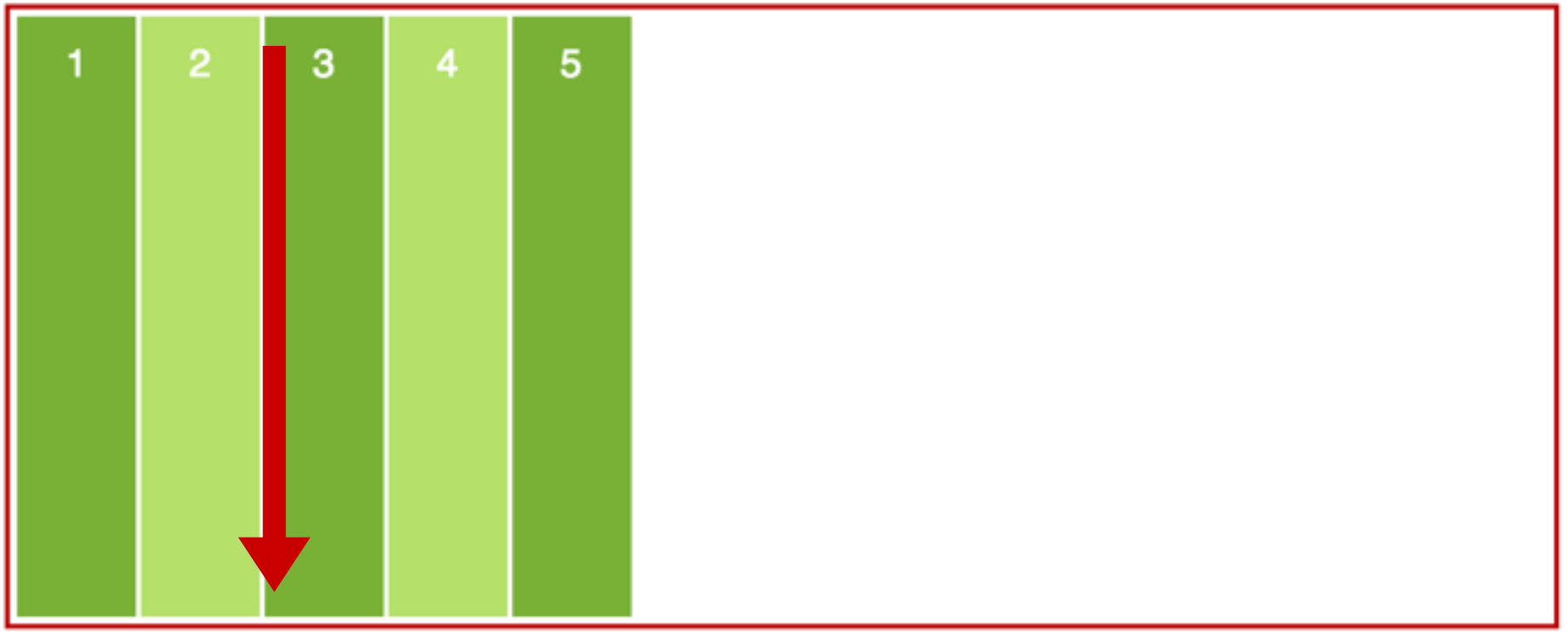
The **align-items property** specifies the default alignment for items inside the flexible container.

```
.parent { align-items: ; }
```

The **stretch value** allows items to be stretched to fit the container. This is the default value.

```
.parent { align-items: stretch; }
```

align-items: stretch



The **center value** allows items to be positioned at the center of the container.

```
.parent { align-items: center; }
```

align-items: centre

The **flex-start value** allows items to be positioned at the beginning of the container.

```
.parent { align-items: flex-start; }
```

align-items: flex-start

The **flex-end value** allows items to be positioned at the end of the container.

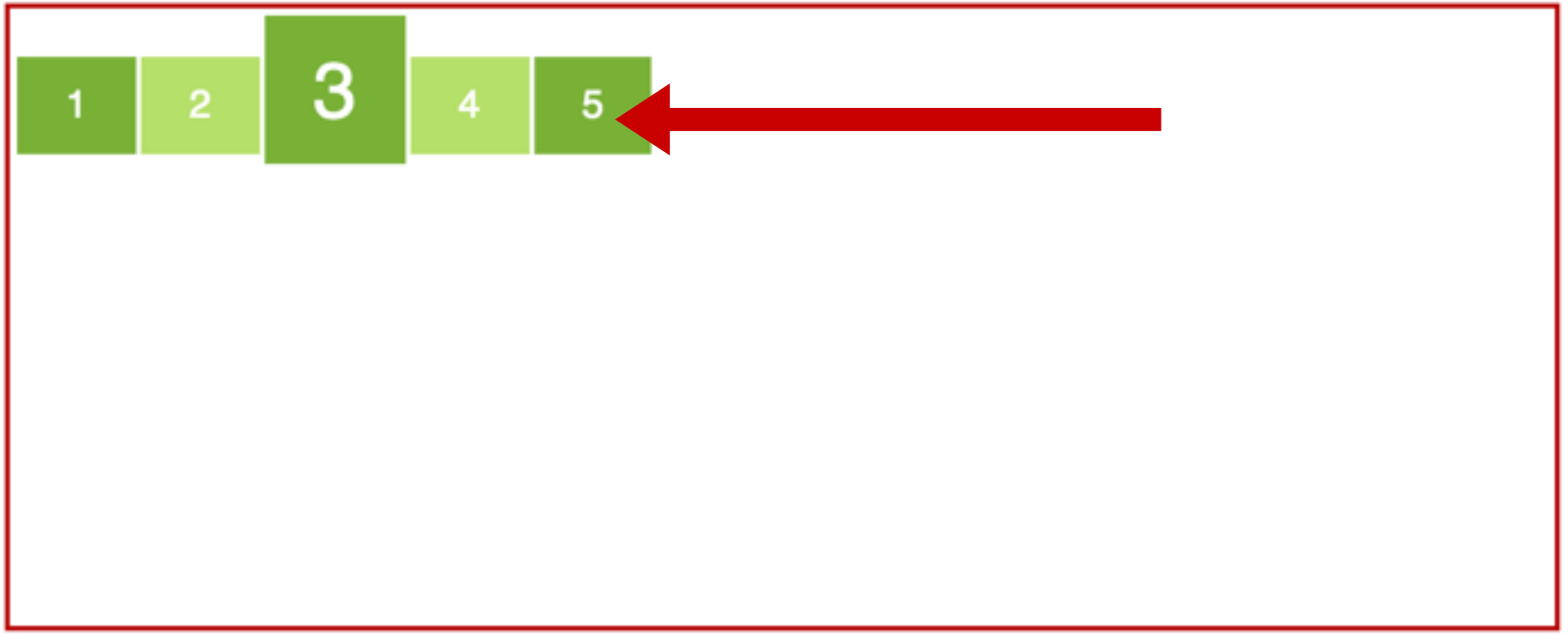
```
.parent { align-items: flex-end; }
```

align-items: flex-end

The **baseline value** allows items to be positioned at the baseline of the container.

```
.parent { align-items: baseline; }
```

align-items: baseline



align-content

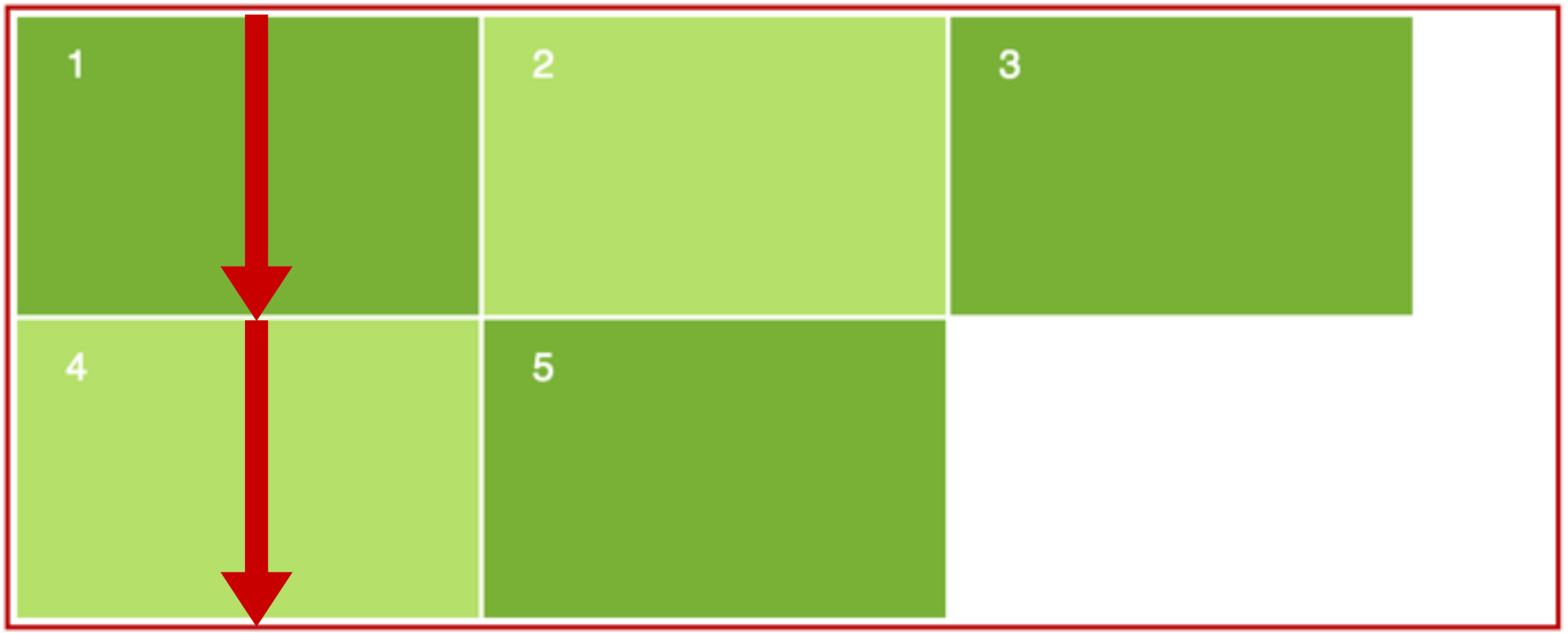
The **align-content property** modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines.

```
.parent { align-content: ; }
```

The **stretch value** allows lines to stretch and take up the remaining space. This is the default value.

```
.parent { align-content: stretch; }
```

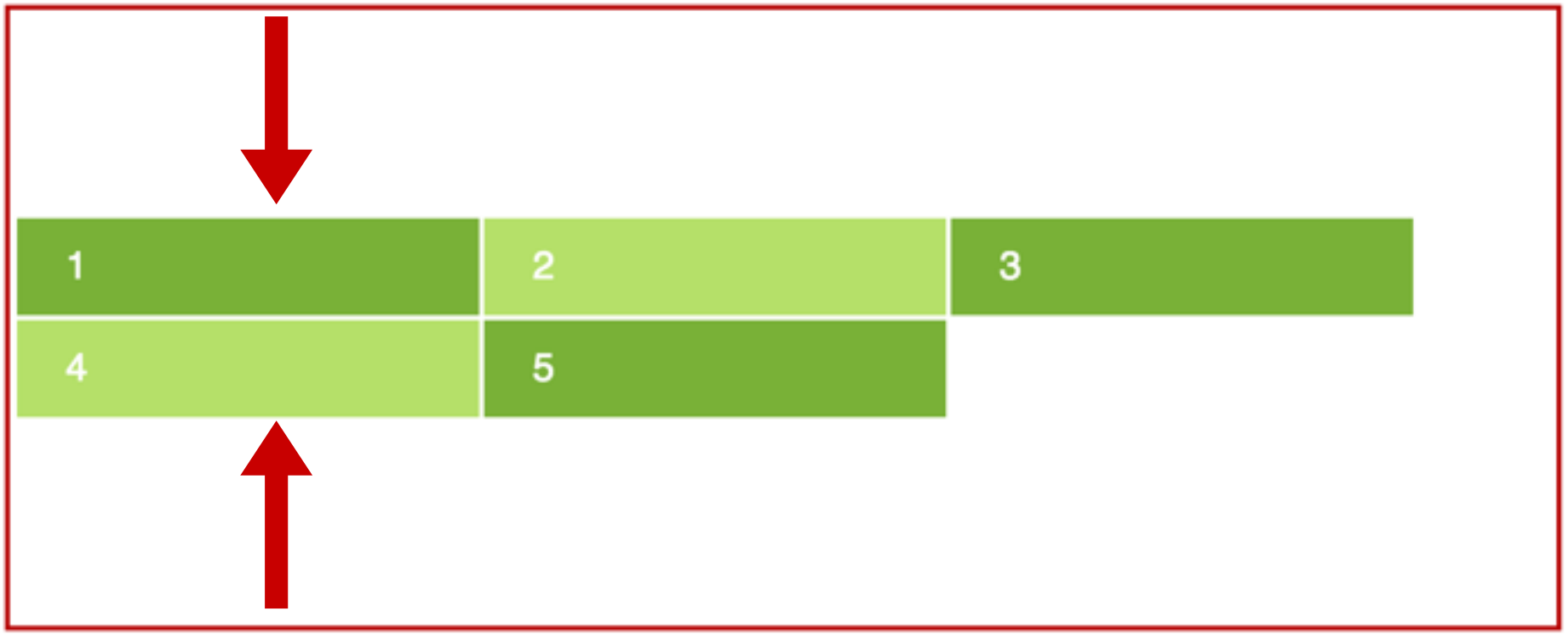
align-content: stretch



The **center value** allows lines to be packed toward the center of the flex container.

```
.parent
{
    flex-wrap: wrap;
    align-content: center;
}
```

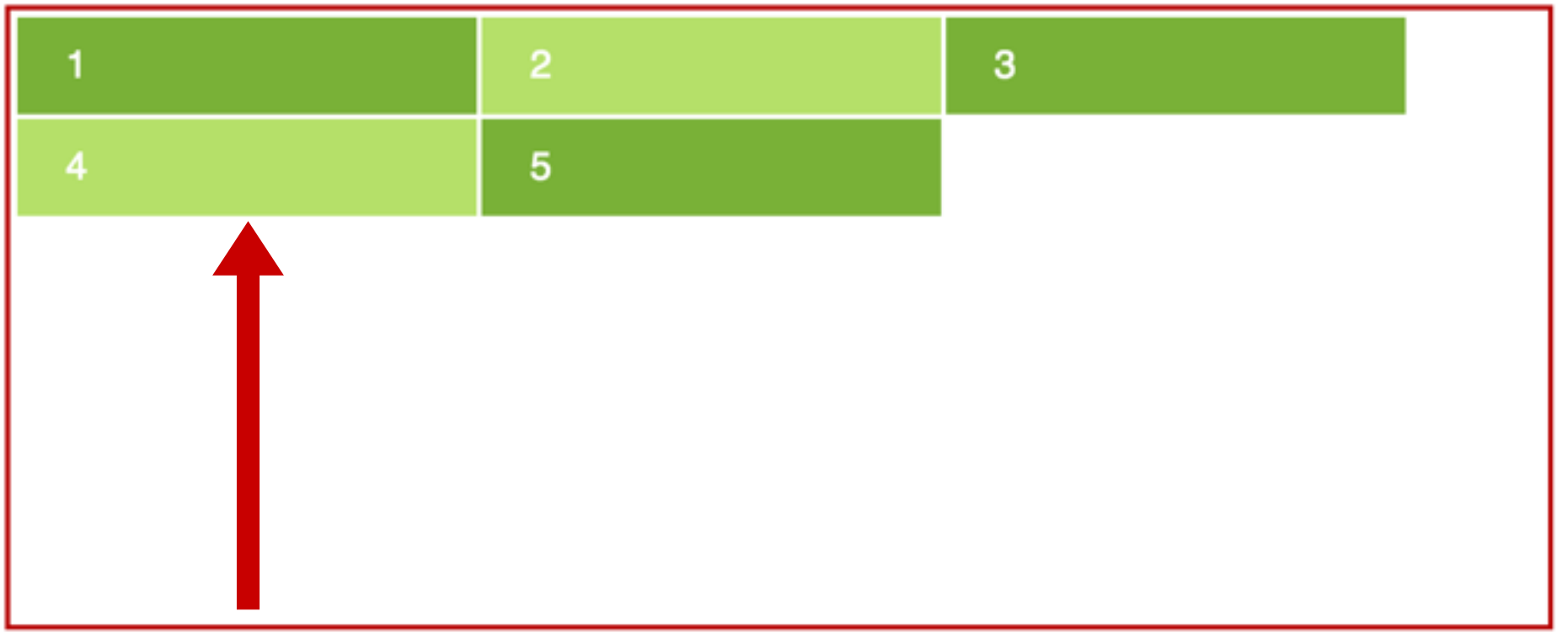
align-content: centre



The **flex-start value** allows lines to be packed toward the start of the flex container.

```
.parent
{
    flex-wrap: wrap;
    align-content: flex-start;
}
```

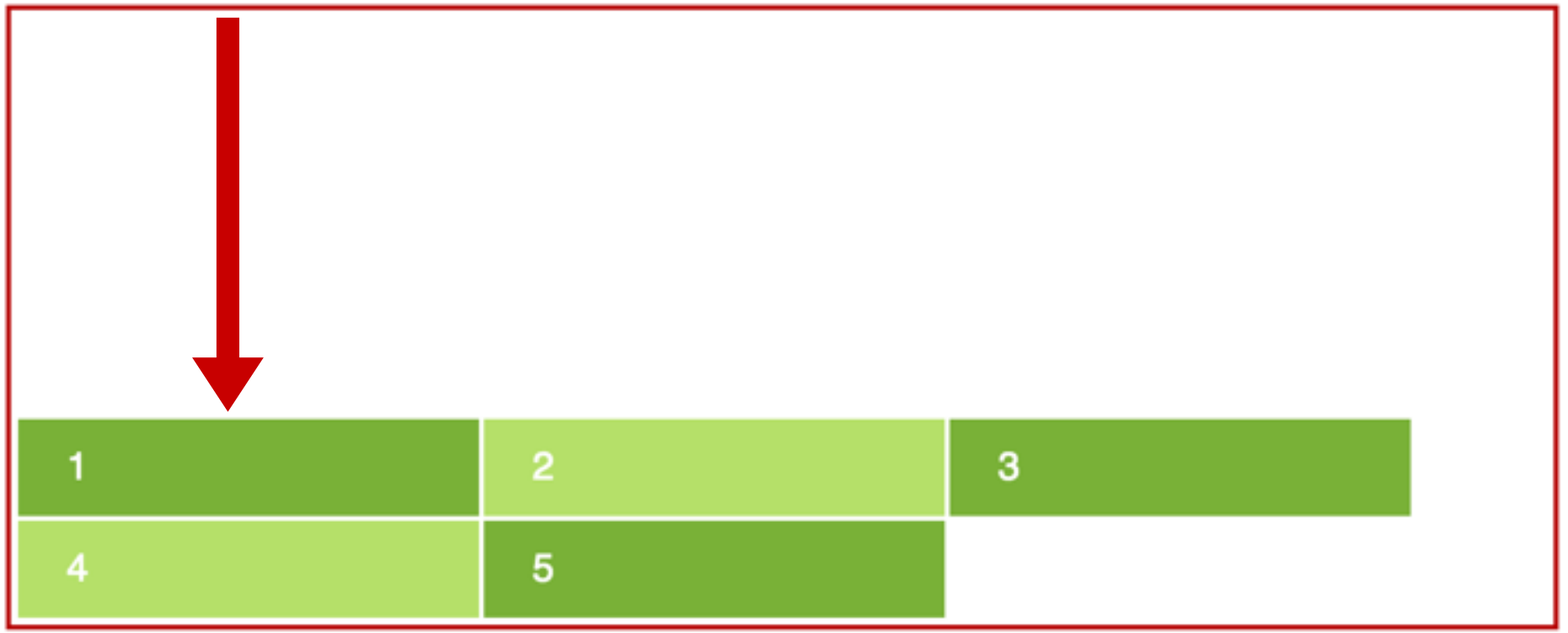
align-content: flex-start



The **flex-end value** allows lines to be packed toward the end of the flex container.

```
.parent
{
    flex-wrap: wrap;
    align-content: flex-end;
}
```

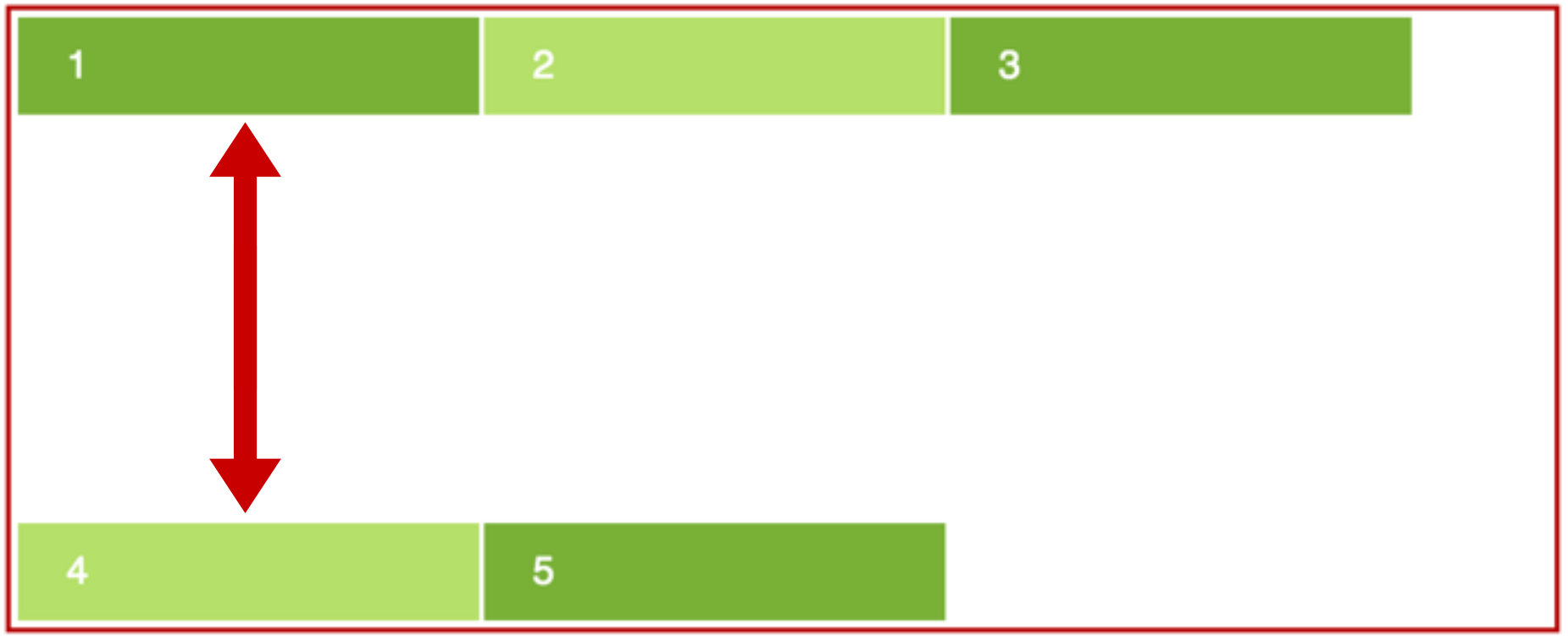
align-content: flex-end



The **space-between value** allows lines to be evenly distributed in the flex container.

```
.parent
{
    flex-wrap: wrap;
    align-content: space-between;
}
```

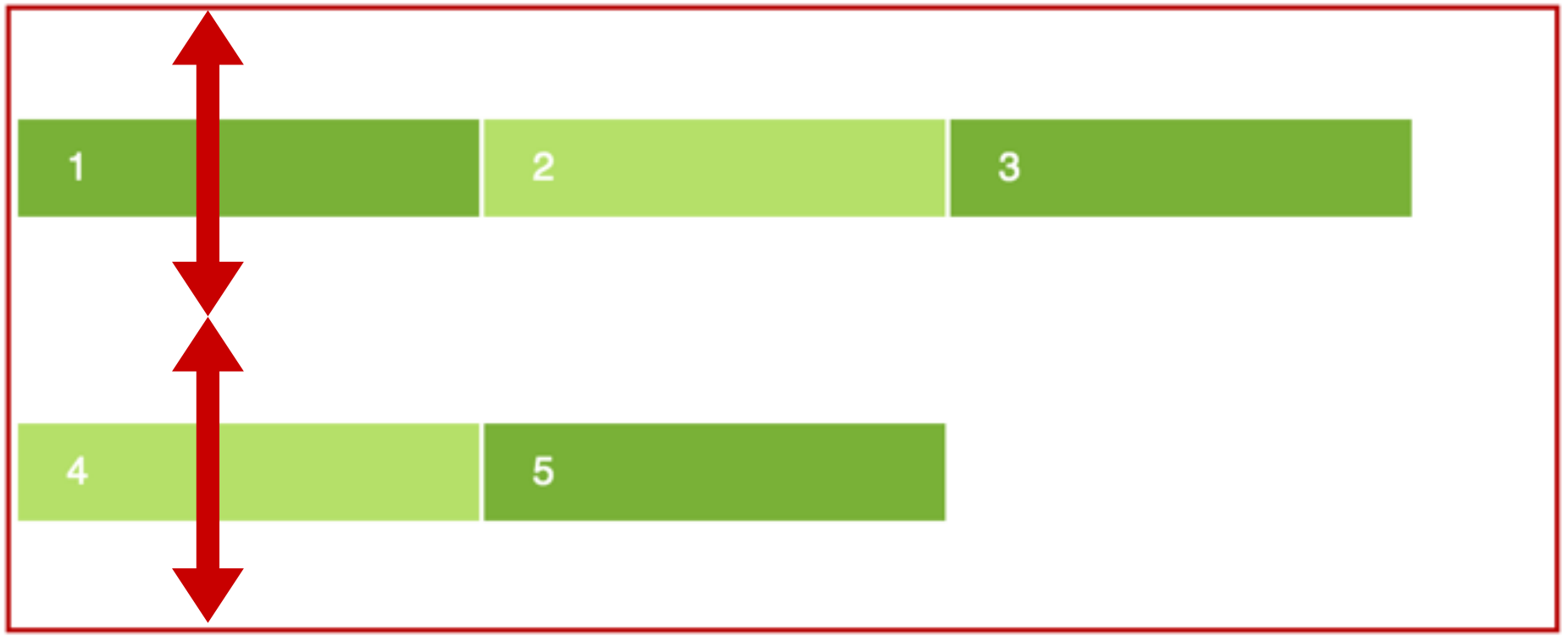
align-content: space-between



The **space-around value** allows lines to be evenly distributed in the flex container, with half-size spaces on either end.

```
.parent
{
    flex-wrap: wrap;
    align-content: space-around;
}
```

align-content: space-around



Properties for the child items

order

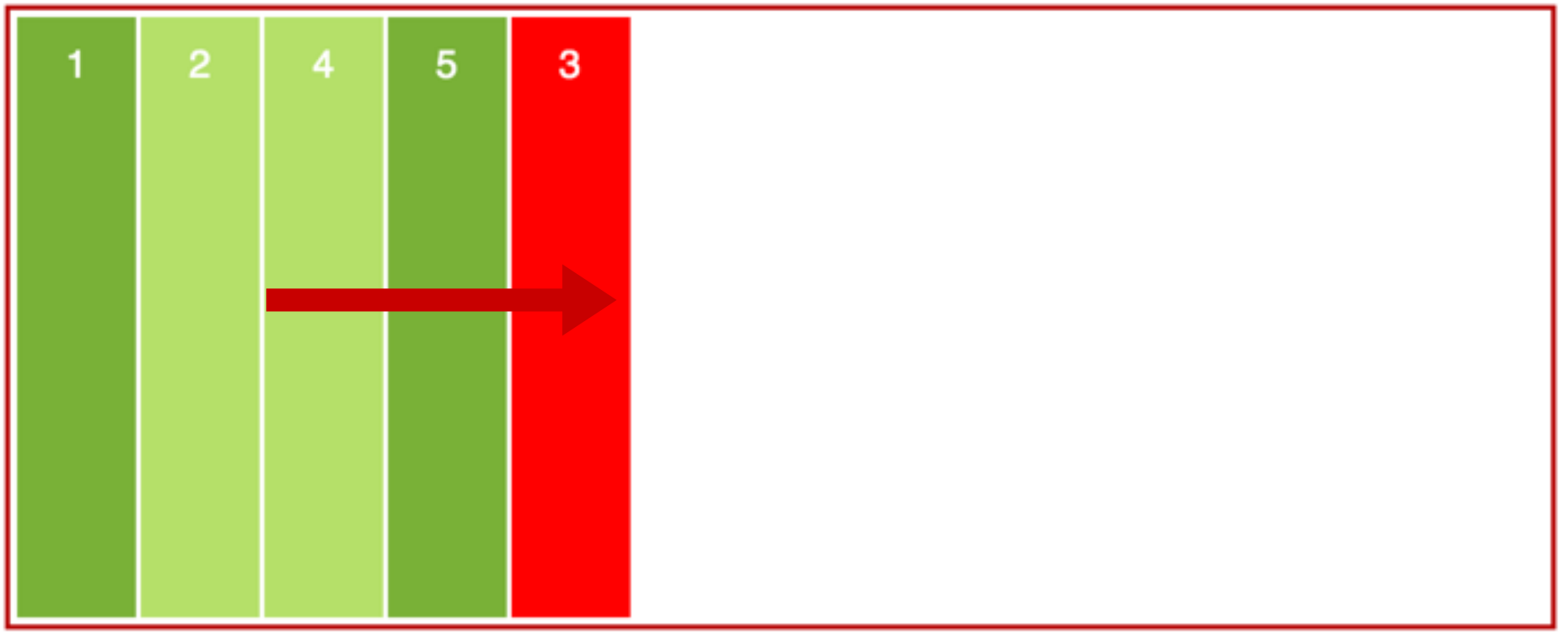
The **order property** specifies the order of a flexible item relative to the rest of the flexible items inside the same container.

```
.item { order: n; }
```

All items have a default value of “0”. If you give an item **a value of “1”** it will move later in order - after all other “0” items.

```
.item { order: 1; }
```

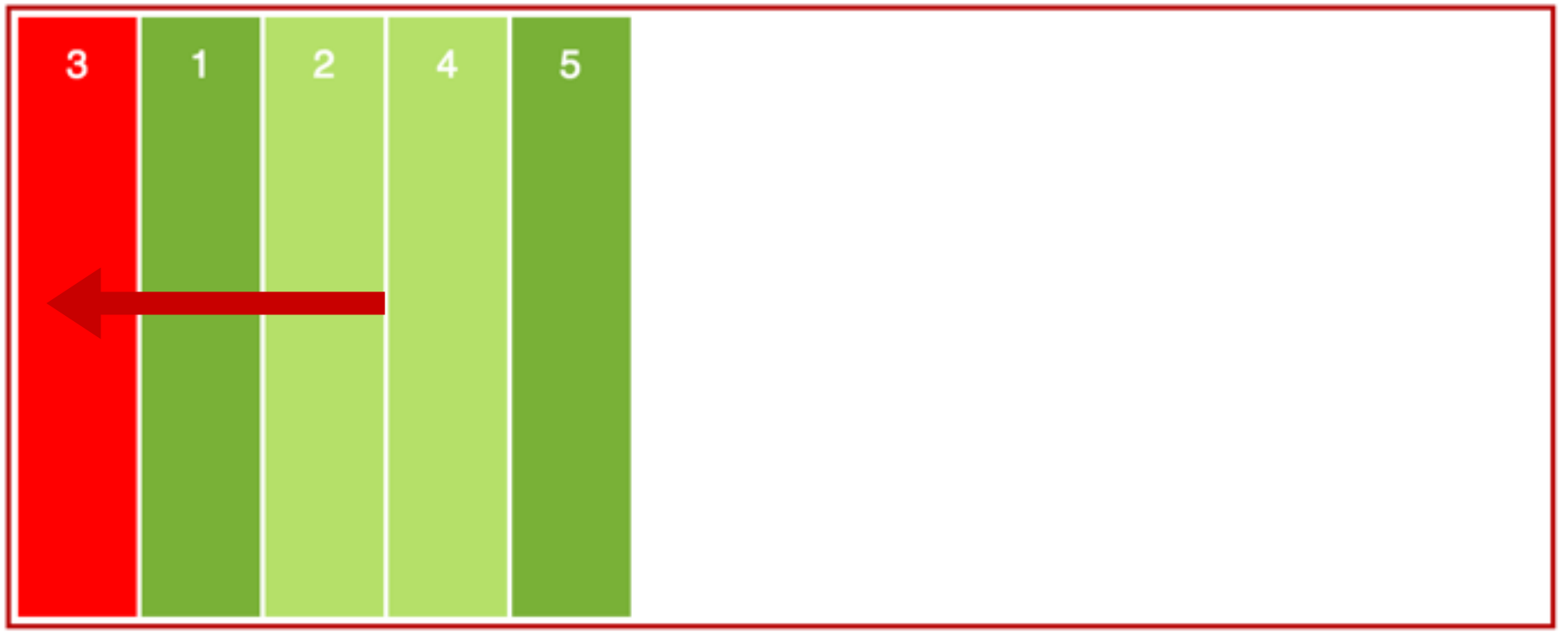
order: 1



If you give an item **a value of “-1”** it will move earlier in order - before all other “0” items.

```
.item { order: 2; }
```

order: -1



flex-grow

The **flex-grow property** specifies how much the item will grow relative to the rest of the flexible items inside the same container. It accepts non-negative unitless values that serves as a proportion.

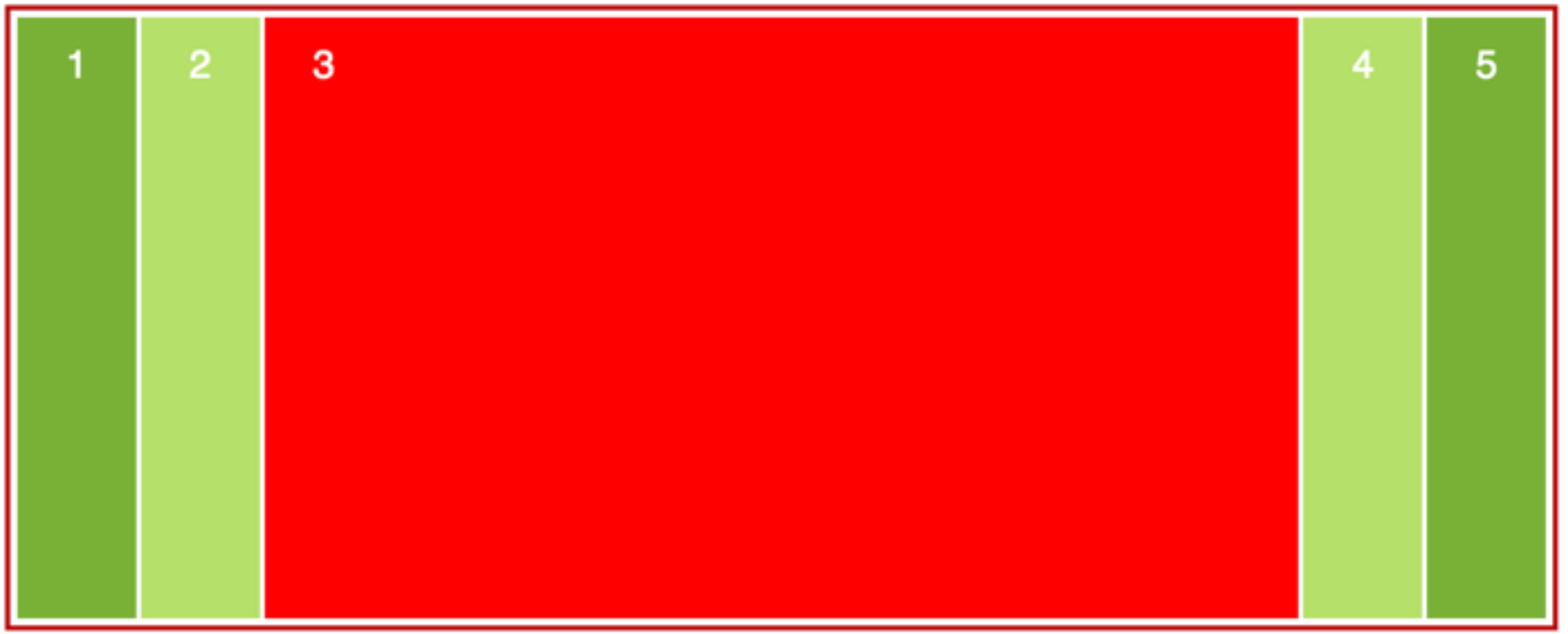
```
.item { flex-grow: ; }
```

If none of the items have flex-grow set, they will effectively all have the **default flex-grow value of “0”**.

However, if one of the items has a value of 1 or above, it will **spread to fill all remaining area** (apart from what has been taken up by the other items).

```
.item { flex-grow: 0; }
.item:nth-child(3) { flex-grow: 1; }
```

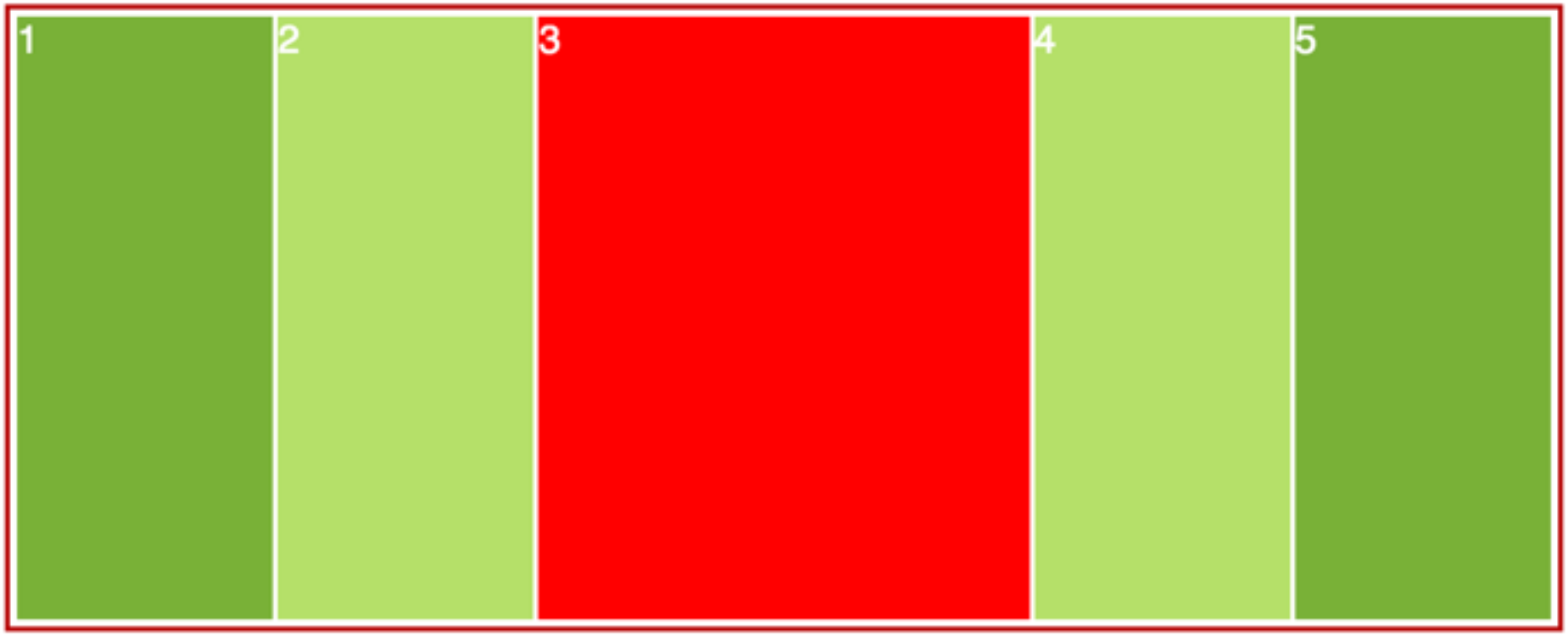
flex-grow: 0 on all items, flex-grow: 1 on red item



If all items have flex-grow set to 1, the remaining space in the container will be distributed equally to all items. If one of the items has a value of 2, it will **take up around twice the remaining space**.

```
.item { flex-grow: 1; }
.item:nth-child(3) { flex-grow: 2; }
```

flex-grow: 1 on all items, flex-grow: 2 on red item



Authors are encouraged to control flexibility using the **flex shorthand property** rather than with flex-grow directly, as the shorthand correctly resets any unspecified components to accommodate common uses.

flex-shrink

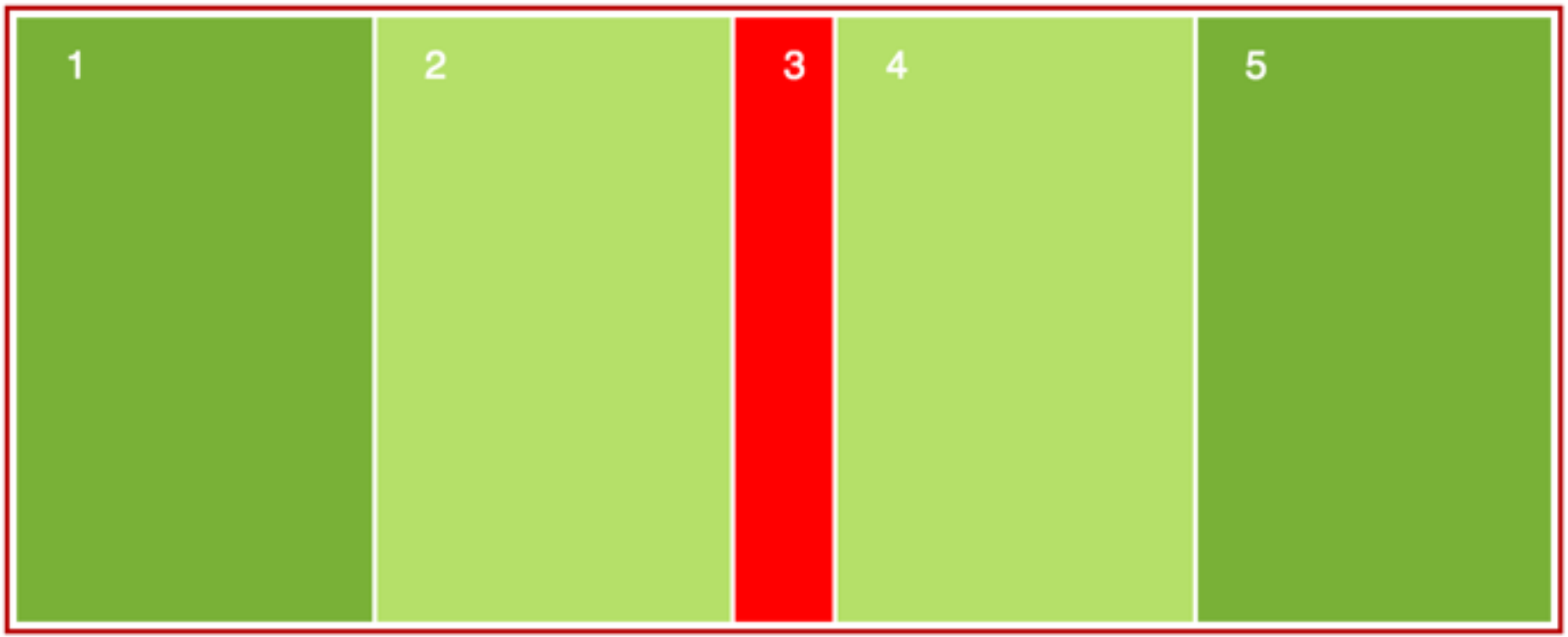
The **flex-shrink property** specifies how the item will shrink relative to the rest of the flexible items inside the same container.

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

The **number value** specifies how much the item will shrink relative to the rest of the flexible items.

```
.item
{
    flex-grow: 2;
    flex-shrink: 0;
}

.item:nth-child(3)
{
    flex-grow: 0;
    flex-shrink: 1;
}
```

flex-grow: 2 on all items, flex-shrink: 1 on red item



Authors are encouraged to control flexibility using the **flex shorthand property** rather than with flex-shrink directly, as the shorthand correctly resets any unspecified components to accommodate common uses.

flex-basis

The **flex-basis property** specifies the initial length of a flexible item before the remaining space is distributed.

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

The **auto value** is equal to the length of the flexible item. If the item has no length specified, the length will be according to its content. This is the default value.

```
.item { flex-basis: auto; }
```

The **number value** can be a length unit, or percentage specifying the initial length of the flexible item(s).

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

The **content keyword value** allows the item to be automatically sized, based on the item's content.

```
.item { flex-basis: content; }
```

align-self

The **align-self property** specifies the alignment for the selected item inside the flexible container.

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

The **auto value** inherits its parent container's align-items property, or "stretch" if it has no parent container.

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

align-self: auto

The **stretch value** allows the element to stretch to fit the container.

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

align-self: stretch

The **center value** allows the element to be positioned at the center of the container.

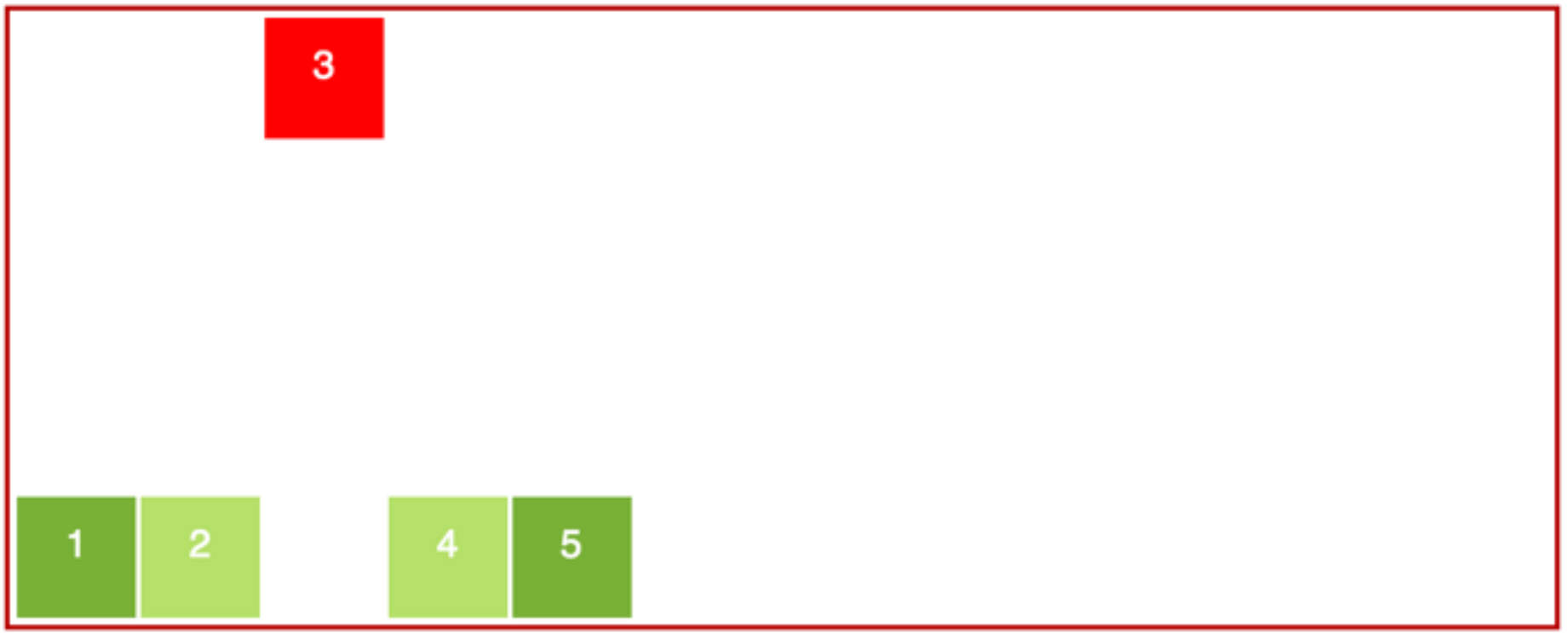
```
.item { align-self: center; }
```

align-self: center

The **flex-start value** allows the element be be positioned at the beginning of the container.

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

align-self: flex-start



The **flex-end value** allows the element to be positioned at the end of the container.

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

align-self: flex-end

The **baseline value** allows the element to be positioned at the baseline of the container.

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

align-self: baseline

flex shorthand

The **flex property** is the shorthand for three properties - flex-grow, flex-shrink and flex-basis. The flex-shrink and flex-basis properties are optional.

```
.item { flex: ; }
```

The **default value** is “0 1 auto”.

```
/* this is the same... */
.item
{
    flex-grow: 0;
    flex-shrink: 1;
    flex-basis: auto;
}

/* as this */
.item { flex: 0 1 auto; }
```

The **none value** is the same as “0 0 auto”.

```
/* this is the same... */
.item
{
    flex-grow: 0;
    flex-shrink: 0;
    flex-basis: auto;
}

/* as this */
.item { flex: 0 0 auto; }

/* and this */
.item { flex: none; }
```

The **auto value** is the same as “1 1 auto”.

```
/* this is the same... */
.item
{
    flex-grow: 1;
    flex-shrink: 1;
    flex-basis: auto;
}

/* as this */
.item { flex: 1 1 auto; }

/* and this */
.item { flex: auto; }
```

Or, the **various values** can be used.

```
.item { flex: <'flex-grow'>; }

.item { flex: <'flex-grow'> <'flex-
shrink'>; }

.item { flex: <'flex-grow'> <'flex-shrink'>
<'flex-basis'>; }
```

Browser support

Flexible Box Layout Module - WD

Global	83.21% + 12.2% =	95.4%
unprefixed:	79.62% + 0.17% =	79.79%

Method of positioning elements in horizontal or vertical stacks. Support includes the support for the all properties prefixed with `flex` as well as `display: flex`, `display: inline-flex`, `align-content`, `align-items`, `align-self`, `justify-content` and `order`.

Current aligned | Usage relative | Show all

IE	Edge *	Firefox	Chrome	Safari	Opera	iOS Safari *	Opera Mini *	Android Browser *	Chrome for Android
								4.1	
8			43					4.3	
9		40	44					4.4	
10		41	45	8		8.4		4.4.4	
11	12	42	46	9	32	9.1	8	44	46
	13	43	47		33				
		44	48		34				
		45	49						

Russ Weakley

Max Design

Site: maxdesign.com.au

Twitter: twitter.com/russmaxdesign

Slideshare: slideshare.net/maxdesign

Linkedin: linkedin.com/in/russweakley