Web Essentials

erasmus
HOGESCHOOL BRUSSEL

SEMANTISCHE
PAGINA
STRUCTUUR

HTML Structuurelementen

(recap)

Semantische betekenis

- `header`
- `footer`
- `nav`
- `main`

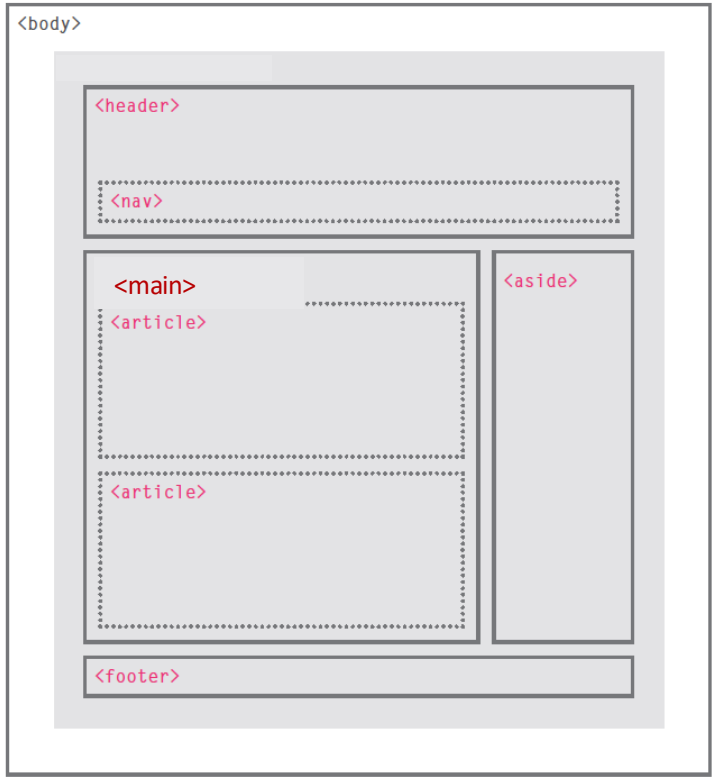
- `article`
- `section`
- `aside`

- `figure`
- geschikt html element
  - h2, label, `blockquote`
  - ….

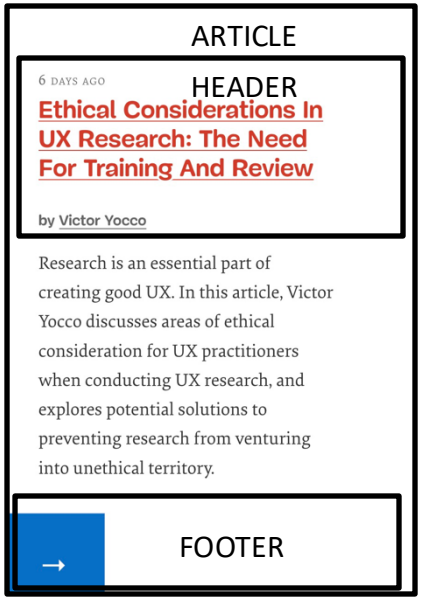
Enkel voor CSS layout

- `div`
- `span`

Op pagina niveau



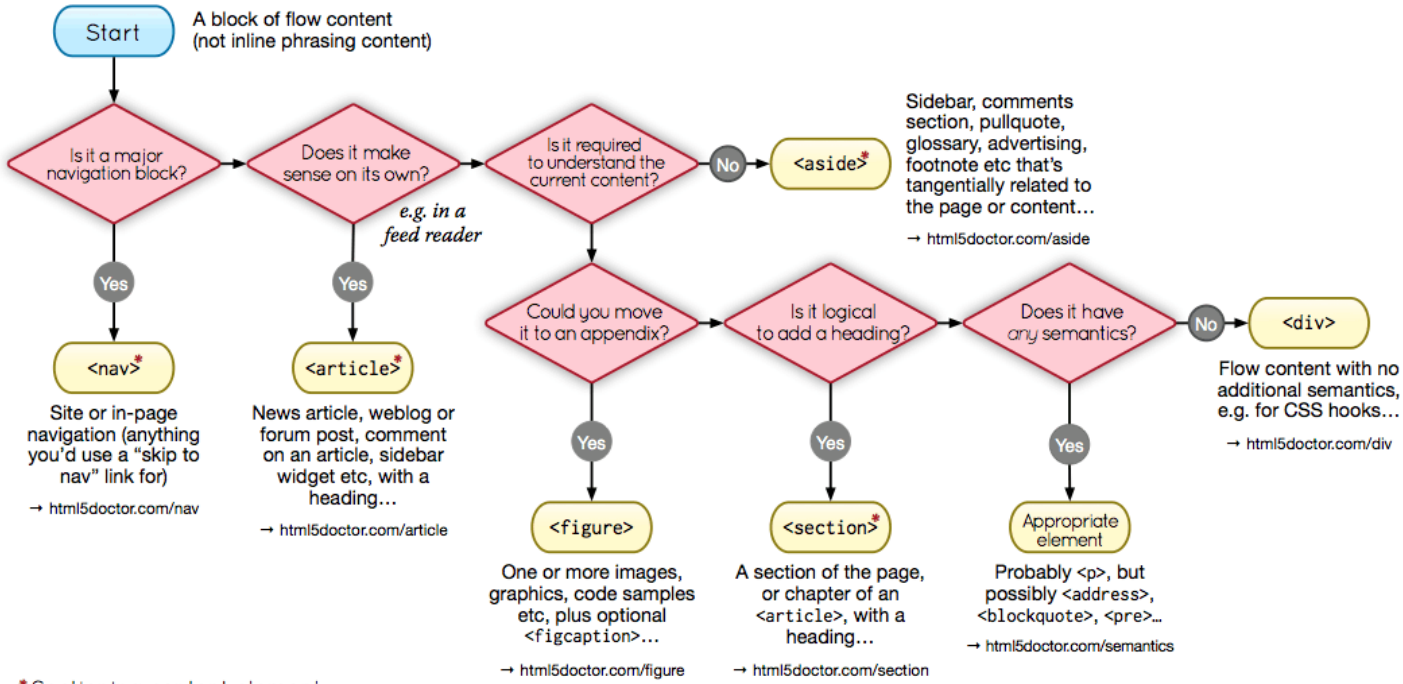
op een lager level



HTML5 Element Flowchart

Sectioning content elements and friends

By @riddle & @boblet
www.html5doctor.com



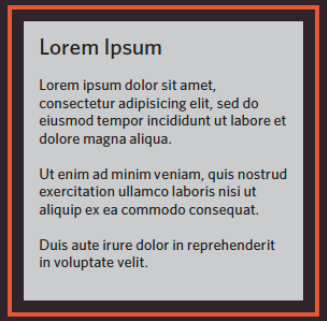
*Sectioning content element

These four elements (and their headings) are used by HTML5's outlining algorithm to make the document's outline
→ html5doctor.com/outline

CSS
POSITIONING

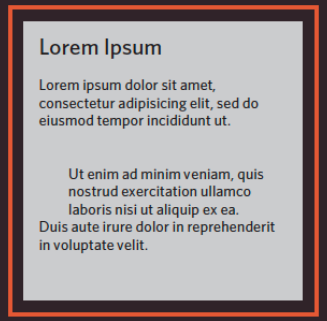
Positioning

static



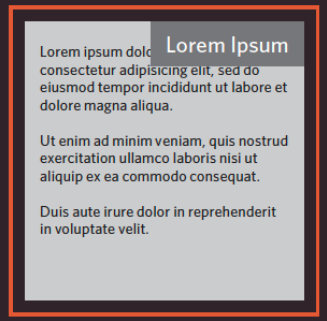
The paragraphs appear one after the other, vertically down the page.

relative



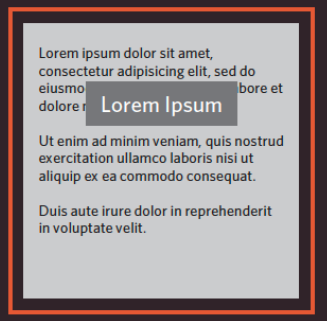
The second paragraph has been pushed down and right from where it would otherwise have been in normal flow.

absolute



The heading is positioned to the top right, and the paragraphs start at the top of the screen (as if the heading were not there).

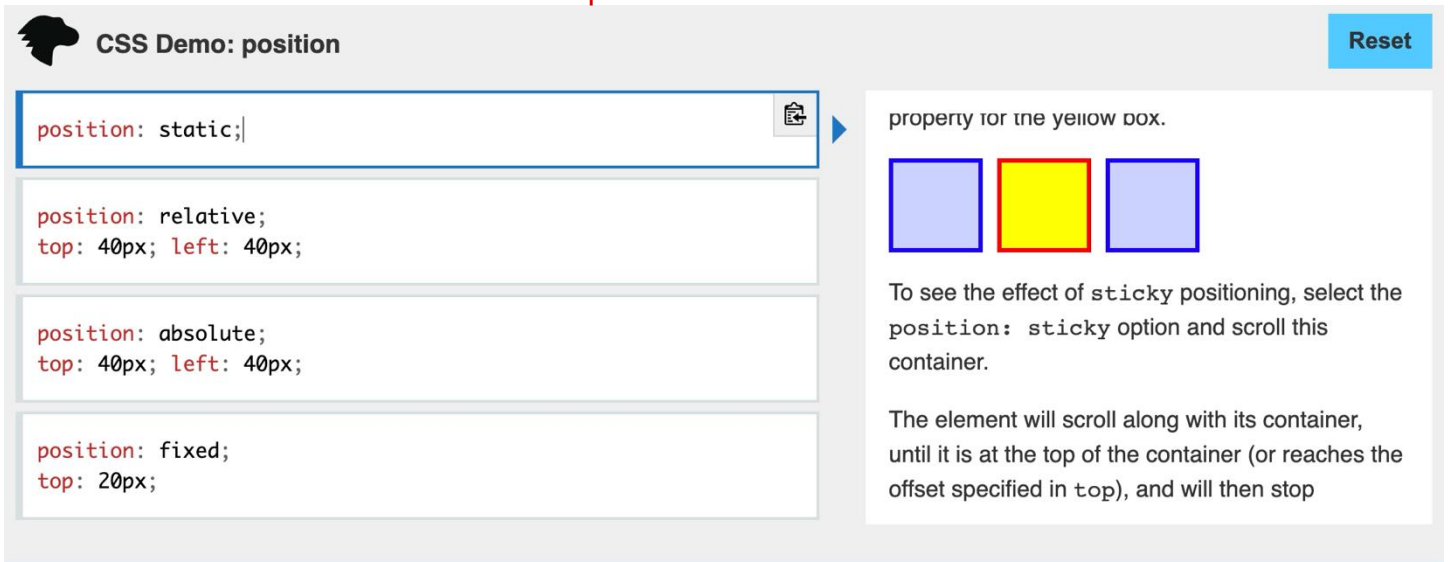
fixed



The heading has been placed in the center of the page and 25% from the top of the screen. (The rest appears in normal flow.)

POSITION

- position: static
- position: absolute
- position: relative
- position: fixed
- position: sticky ==> Element zijn positie behoudt, zelfs wanneer je scrolt, totdat een bepaald punt is bereikt



POSITION DETAILS

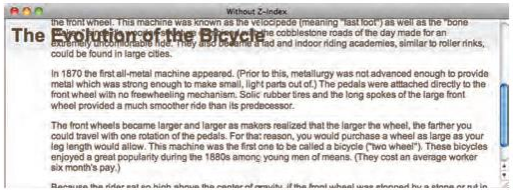
- Offset
  - top: value
  - left: value
  - right: value
  - bottom: value
- width
- height
- stacking order
  - z-index

OVERLAPPENDE ELEMENTEN
Z-INDEX

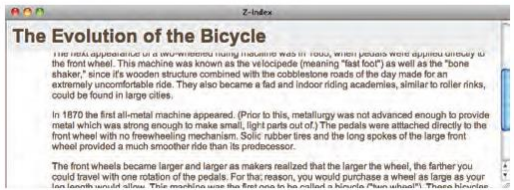
chapter-15/z-index.html CSS

```
h1 {
  position: fixed;
  top: 0px;
  left: 0px;
  margin: 0px;
  padding: 10px;
  width: 100%;
  background-color: #efefef;
  z-index: 10;}
p {
  position: relative;
  top: 70px;
  left: 70px;}
```

RESULT WITHOUT Z-INDEX



RESULT WITH Z-INDEX

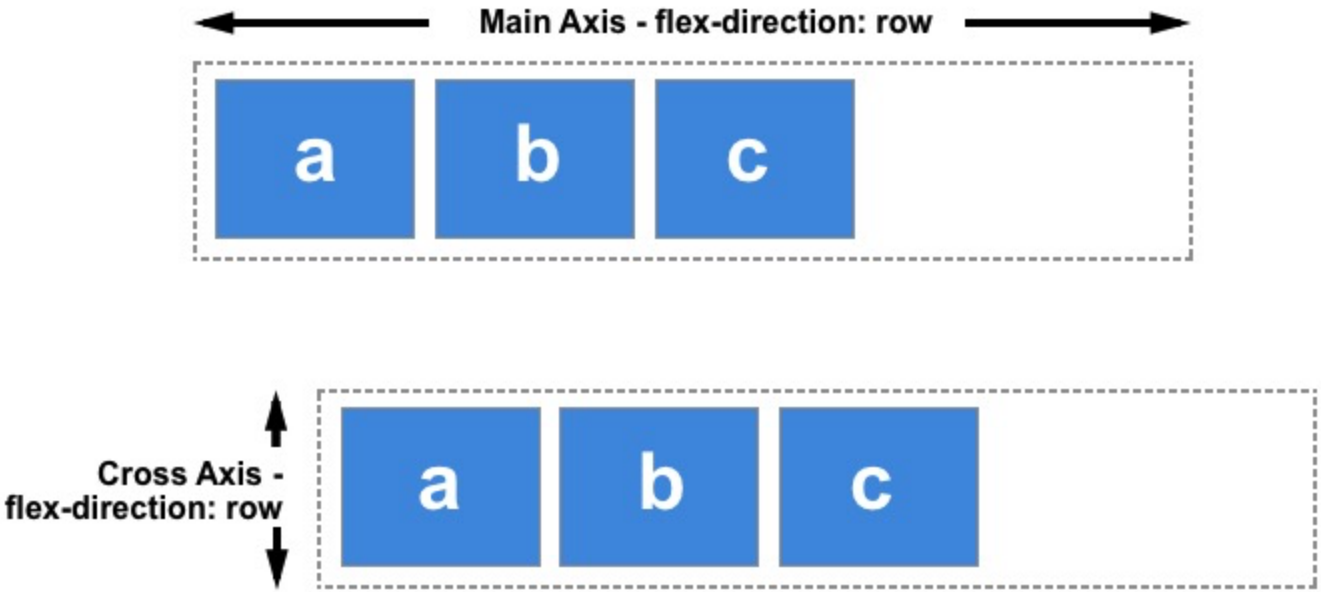


FLEXBOX
LAYOUT

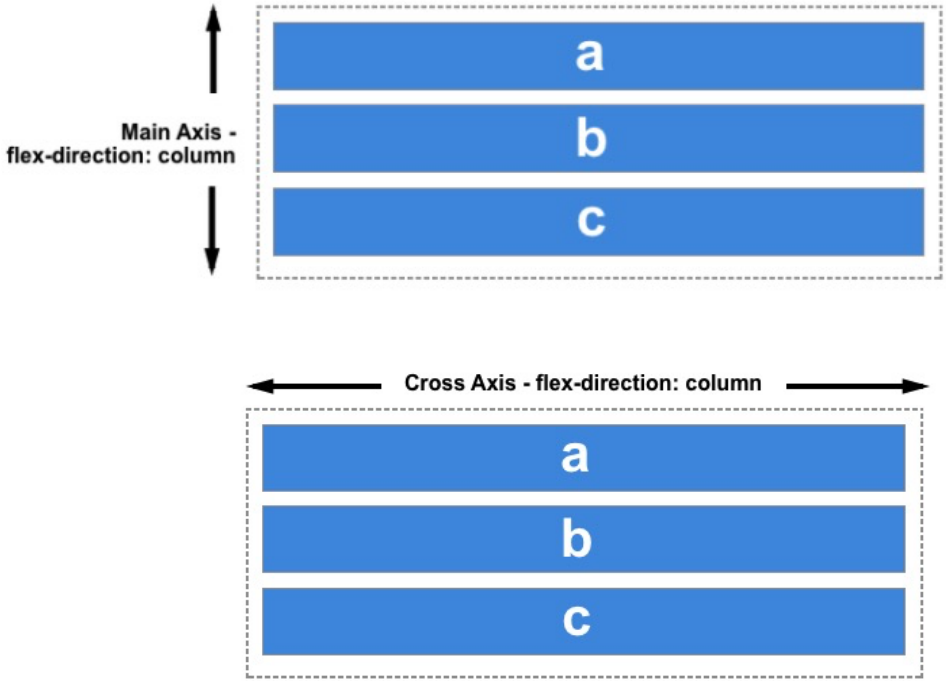
CSS FLEXBOX

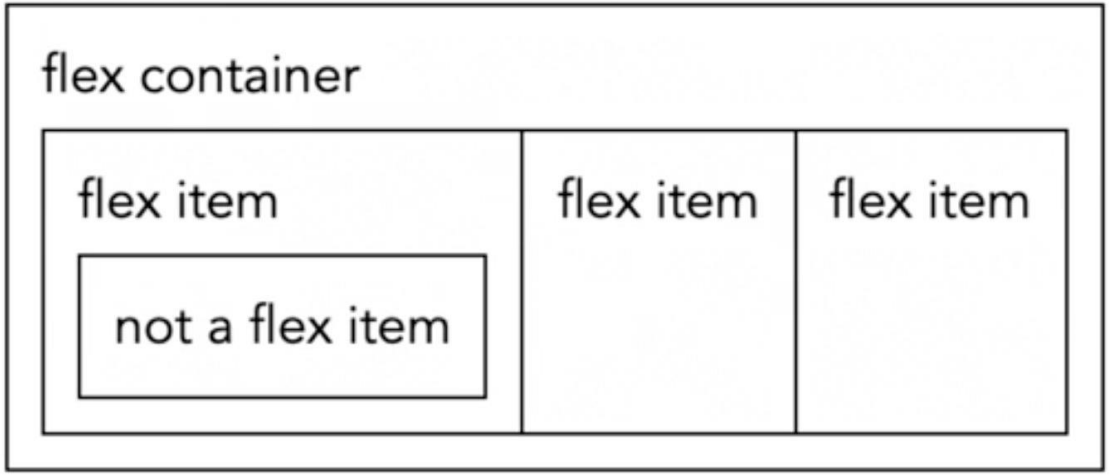
- LAYOUT IN ÉÉN DIMENSIE
  - ofwel horizontaal (row)
  - ofwel vertikaal (column)
- veel controle over uitlijning van inhoud
  - zowel horizontaal & vertikaal
- Eenvoudig om volgorde van inhoud te wijzigen

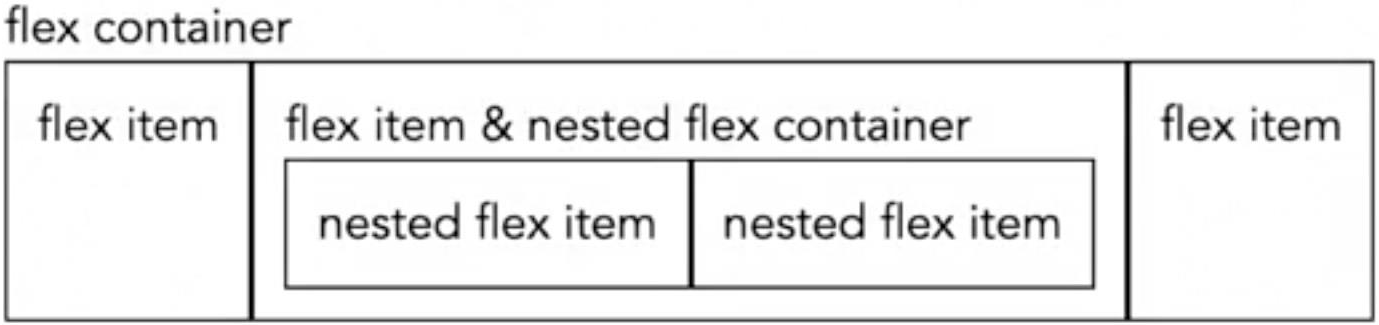
ROW AXIS


Main Axis - flex-direction: row
a
b
c
Cross Axis -
flex-direction: row
a
b
c

COLUMN AXIS


Main Axis -
flex-direction: column
a
b
c
Cross Axis - flex-direction: column
a
b
c


flex container
flex item
not a flex item
flex item
flex item


flex container
flex item
flex item & nested flex container
nested flex item
nested flex item
flex item

FLEXBOX PROPERTIES

FOR THE PARENT

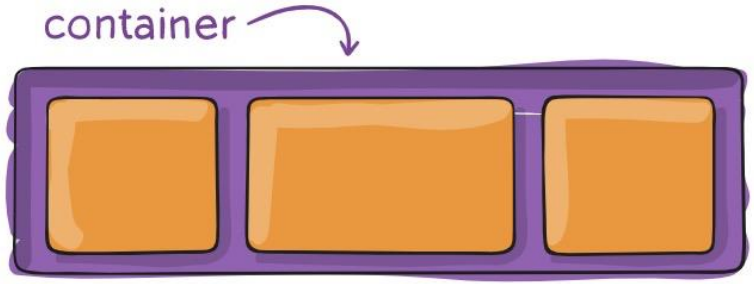
- **display: flex** | inline-flex
- flex-direction
- flex-wrap
- flex-flow
- justify-content
- align-items
- align-content
- **gap**
  row-gap | column-gap

FOR CHILDREN

- flex-grow
- flex-shrink
- flex-basis
- **flex**
- align-self
- order

all visuals in this section © css-tricks.com

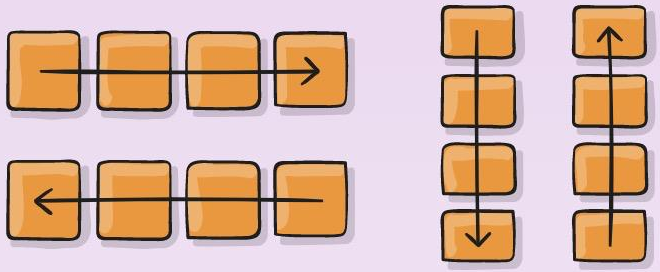
PROPERTIES FOR THE PARENT (flex container)




```
.container {
  display: flex; /* or inline-flex */
}
```

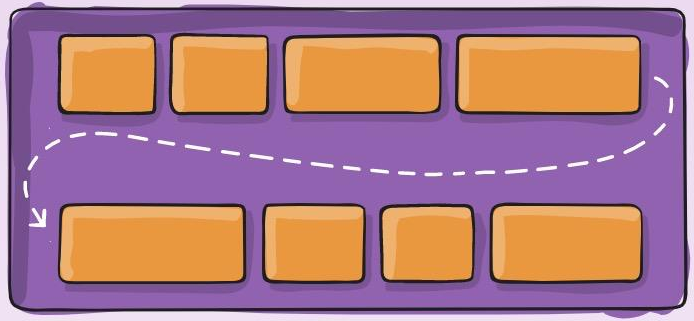
copyright for all images about flexbox:
Chris Coyier - css-tricks.com/snippets/css/a-guide-to-flexbox

flex-direction



```
.container {
  flex-direction: row | row-reverse | column |
column-reverse;
}
```

flex-wrap

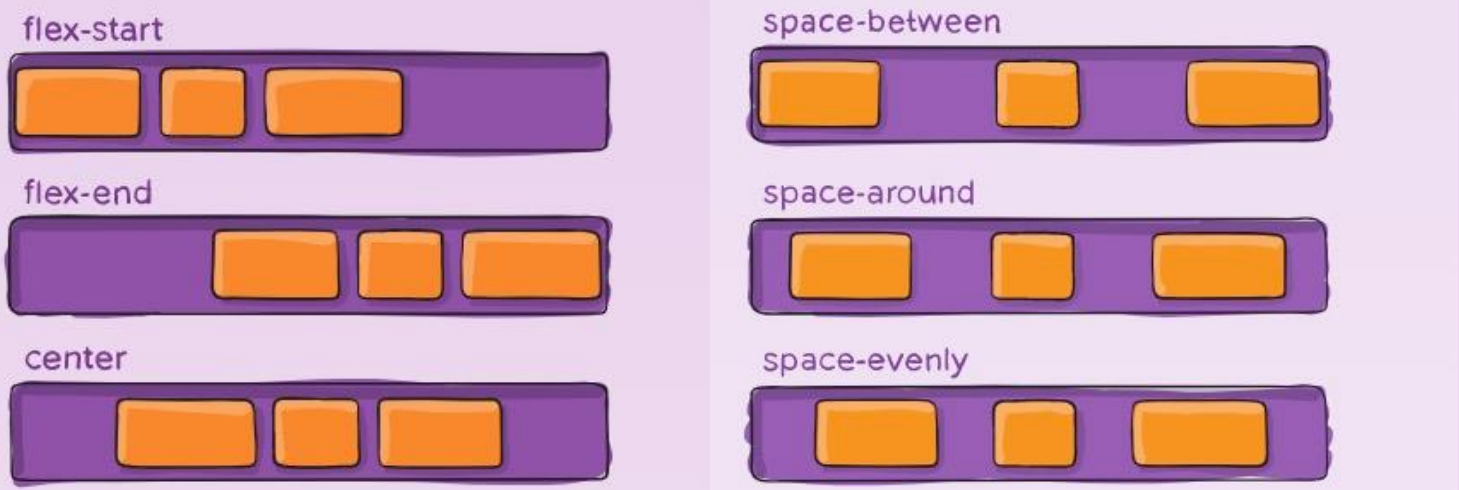


```
.container {
  flex-wrap: nowrap | wrap | wrap-reverse;
}
```

flex-flow

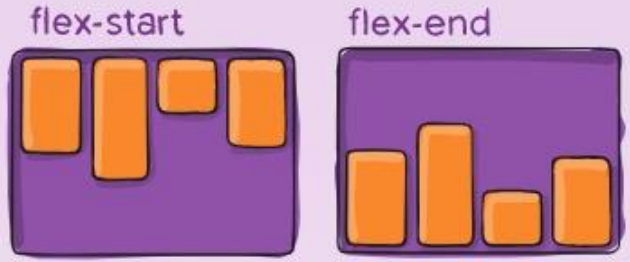
This is a shorthand for the flex-direction and flex-wrap properties, which together define the flex container’s main and cross axes. The default value is row nowrap

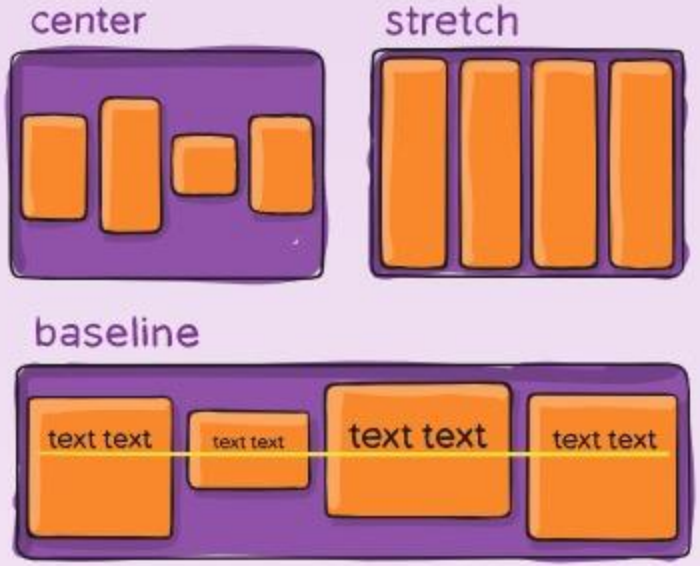
justify-content




```
.container {
  justify-content: flex-start | flex-end | center |
space-between | space-around | space-evenly;
}
```

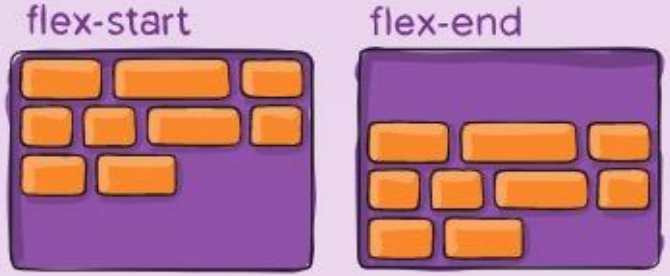
align-items

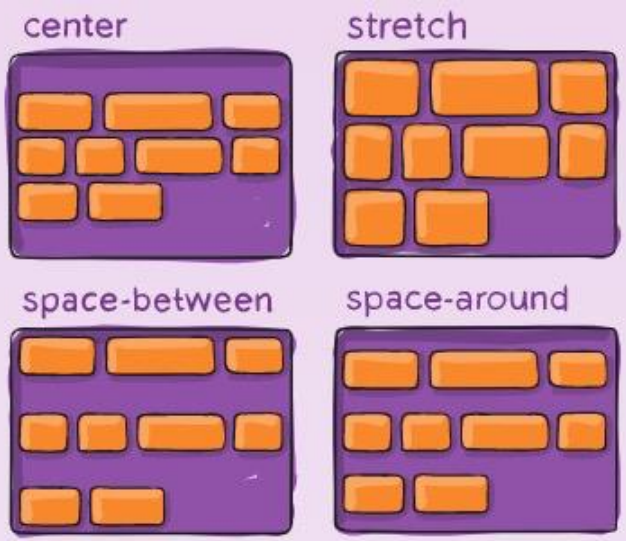





```
.container {
  align-items:  flex-start | flex-end | center |
stretch | baseline;
}
```

align-content



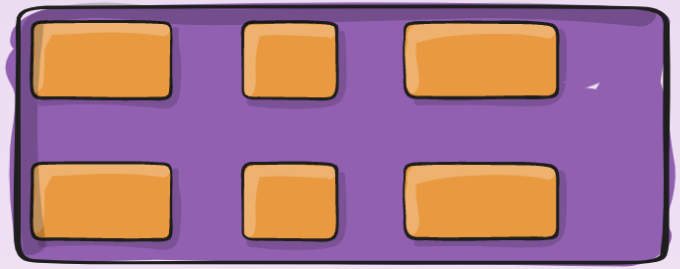



```
.container {
  align-items:  flex-start | flex-end | center |
stretch | space-between | space-around;
}
```

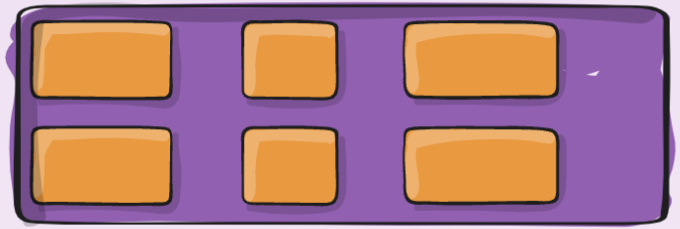
gap, row-gap, column-gap

gap: 10px

gap: 30px



gap: 10px 30px

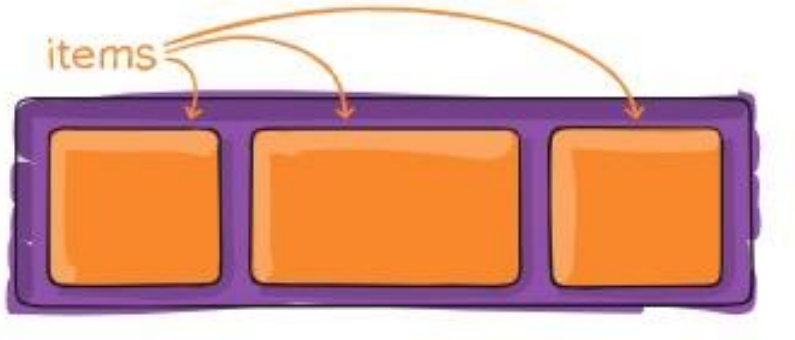


```
.container {
  display: flex;
  ...
  gap: 10px;
  gap: 10px 20px; /* row-gap column gap */
  row-gap: 10px;
  column-gap: 20px;
}
```

Enkel **tussen** de flex-items
niet er rond

Het is eigenlijk een *minimum gap*
als er b.v. justify-content: space-between
is gebruikt kan de werkelijke gap groter zijn

PROPERTIES FOR THE CHILDREN (flex items)



flex-grow

```
.item {
  flex-grow: 4; /* default 0 */
}
```

flex-shrink

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

```
.item {
  flex-shrink: 3; /* default 1 */
}
```

flex-basis

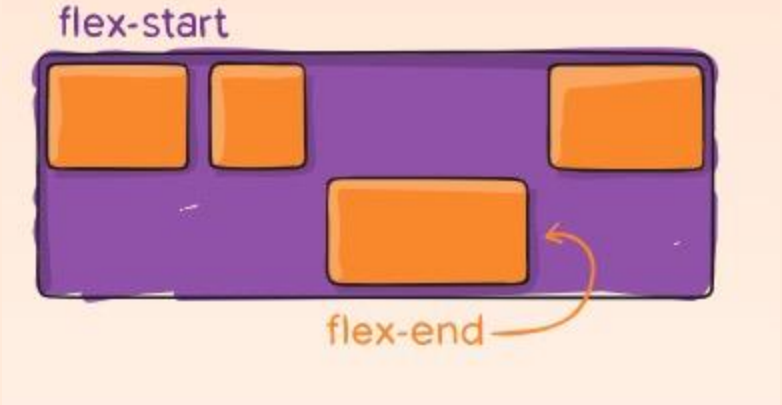
This defines the default size of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword.

```
.item {
  flex-basis: number | auto; /* default auto */
}
```

flex

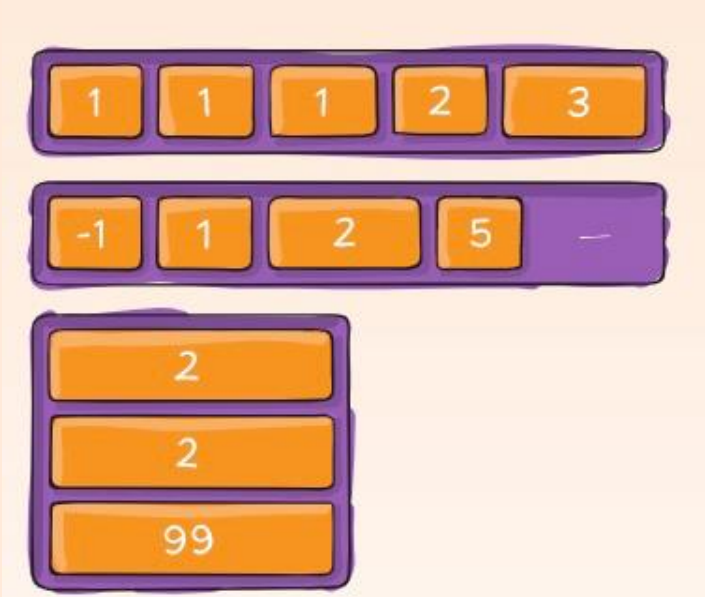
This is the shorthand for `flex-grow`, `flex-shrink` and `flex-basis` combined. The second and third parameters (`flex-shrink` and `flex-basis`) are optional. The default is `0 1 auto` , but if you set it with a single number value, it's like `1 0` .

align-self




```
.item {
  align-self: auto | flex-start | flex-end |
center | baseline | stretch;
}
```

order




```
.item {
  order: 5; /* default is 0 */
}
```

justify-content

flex-start

space-around

center

space-between

flex-end

space- evenly

align-content

flex-start

center

flex-end

space-around

space-between

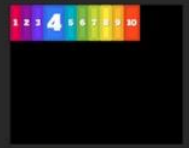
space-evenly

stretch

align-items

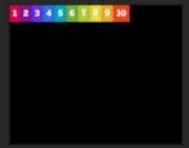
stretch

stretch



baseline

flex-around



center

flex-end

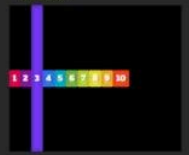
align-self

flex-start

center

flex-end

stretch



baseline

auto

© medium.com/swlh