

# Web Essentials





SEMANTISCHE  
PAGINA  
STRUCTUUR

# HTML Structuurelementen

- div
- header
- footer
- nav
- main
- article
- section
- aside

Dit gaat **niet** over ~~LAYOUT~~  
maar over **semantische structuur**

# header, footer

- Niet alleen op pagina niveau maar ook dieper genest
  - bv: Alle items op een nieuwspagina kunnen een eigen header en footer hebben
- headers & footers mogen niet in andere headers & footers genest zijn.

# header

`<header> ... </header>`

- represents a group of introductory or navigational aids.  
A header element is intended to usually contain the **section's heading** (an h1–h6 element or an hgroup element), but this is not required.  
The header element can also be used to wrap a section's table of contents, a search form, or any relevant logos.

# footer

`<footer> ... </footer>`

- represents a footer for its nearest ancestor sectioning content or sectioning root element. A footer typically contains information about its section such as who wrote it, links to related documents, copyright data, and the like

```
<div class="footer">
  <ul>
    <li>copyright</li>
    <li>sitemap</li>
    <li>contact</li>
    <li>to top</li>
  </ul>
</div>
```

```
<footer>
  <ul>
    <li>copyright</li>
    <li>sitemap</li>
    <li>contact</li>
    <li>to top</li>
  </ul>
</footer>
```

# nav

`<nav> ... </nav>`

- The nav element represents a section of a page that links to other pages or to parts within the page: a section with (major) navigation links

```
<header>
  <h1>Titel hier</h1>
  <nav>
    (ongeordende lijst met) sitelinks
  </nav>
</header>
```

- Niet elke lijst van links moet binnen en nav

# main

`<main> ... </main>`

- The main content of the body of a document or application. The main content area consists of content that is directly related to or expands upon the central topic of a document or central functionality of an application.  
(W3C)



# section

`<section> ... <section>`

- The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content, typically with a heading
- latest work, product list, specifications of a product, 'action' category on a game sites overview page, ...

# article

`<article> ... </article>`

- The article element represents a self-contained composition in a document, page, application, or site and that is, in principle, independently distributable or reusable, e.g. in syndication.
- blog post, product, news item
- Any content that make sense by itself
  - news item, blogpost, tweet

# section - article

## WRONG

```
<div class="news-items">
<div class="post-summary">
<!-- post summary -->
</div>
<div class="post-summary">
<!-- post summary -->
</div>
<!-- etc. -->
</div>
<div class="tweets">
<div class="tweet"> ... </div>
<div class="tweet"> ... </div>
<div class="tweet"> ... </div>
</div>
```

## RIGHT

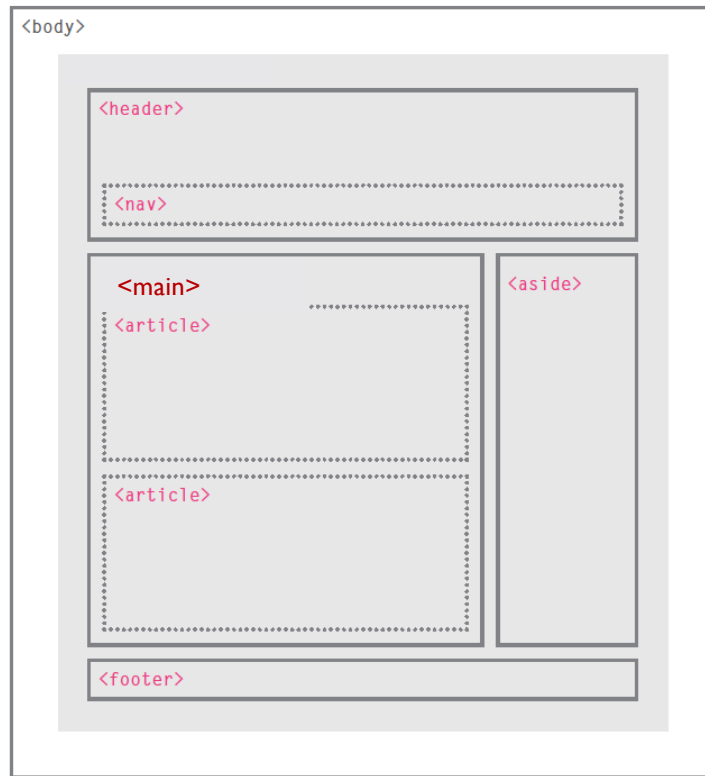
```
<section class="news-items">
<article class="post-summary">
<!-- post summary -->
</article>
<article class="post-summary">
<!-- post summary -->
</article>
<!-- etc. -->
</section>
<section class="tweets">
<article> ... </article>
<article> ... </article>
<article> ... </article>
</section>
```

# aside

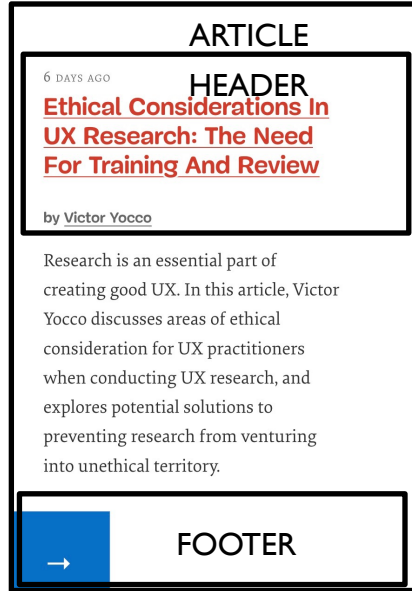
`<aside> ... </aside>`

- The aside element represents a section of a page that consists of content that is tangentially related to the content around the aside element, and which could be considered separate from that content.
- testimonials, related products, downloads for a product, pull quote, comments

# Op pagina niveau



# op een lager level



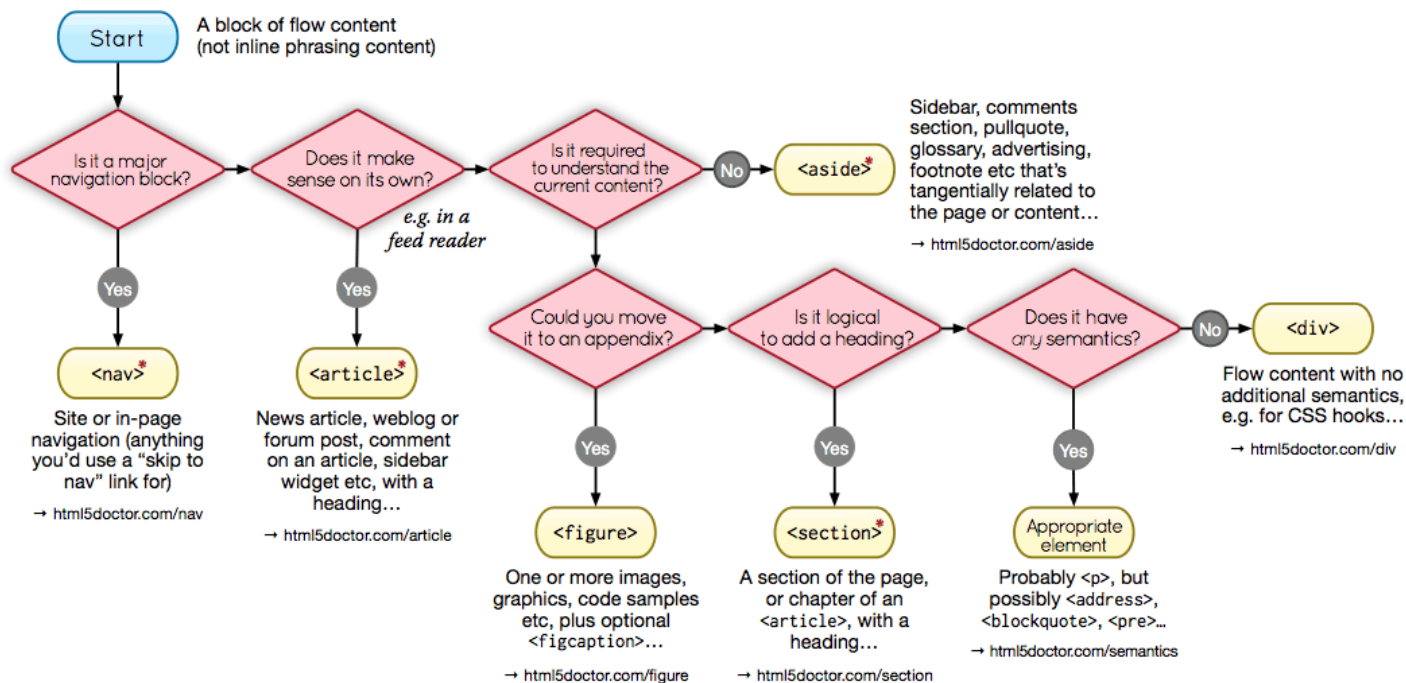


Doctor

# HTML5 Element Flowchart

Sectioning content elements and friends

By @riddle & @boblet  
[www.html5doctor.com](http://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](http://html5doctor.com/outline)

2011-07-22 v1.5  
For more information:  
[www.html5doctor.com/semantics](http://www.html5doctor.com/semantics)

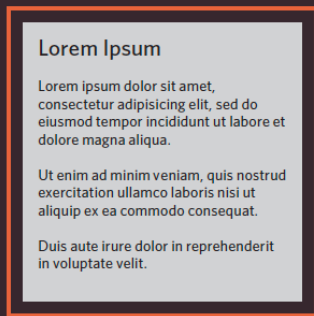


# CSS POSITIONING



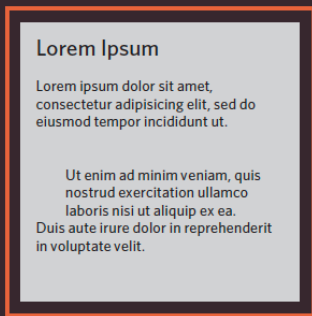
# Positioning

## static



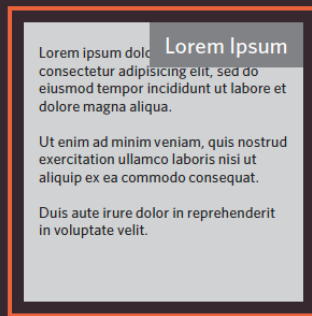
The paragraphs appear one after the other, vertically down the page.

## relative



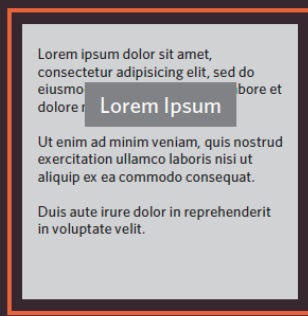
The second paragraph has been pushed down and right from where it would otherwise have been in normal flow.

## 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



## CSS Demo: position

[Reset](#)

```
position: static;
```

```
position: relative;  
top: 40px; left: 40px;
```

```
position: absolute;  
top: 40px; left: 40px;
```

```
position: fixed;  
top: 20px;
```



property for the yellow box.



To see the effect of `sticky` positioning, select the `position: sticky` option and scroll this container.

The element will scroll along with its container, until it is at the top of the container (or reaches the offset specified in `top`), and will then stop

# POSITION DETAILS

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

# OVERLAPPING ELEMENTS

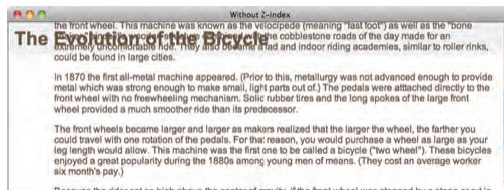
## Z-INDEX

chapter-15/z-index.html

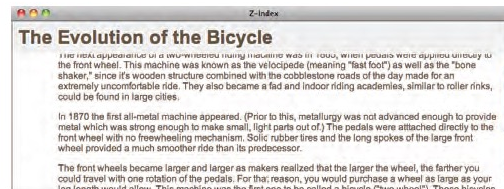
CSS

```
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



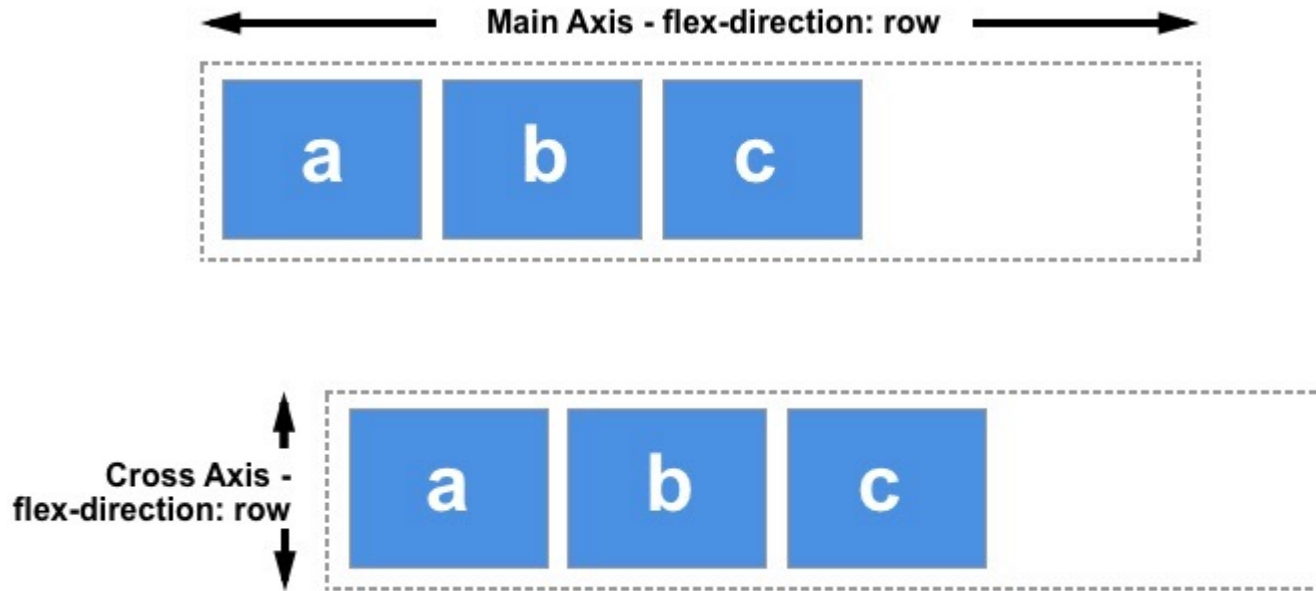


# FLEXBOX LAYOUT

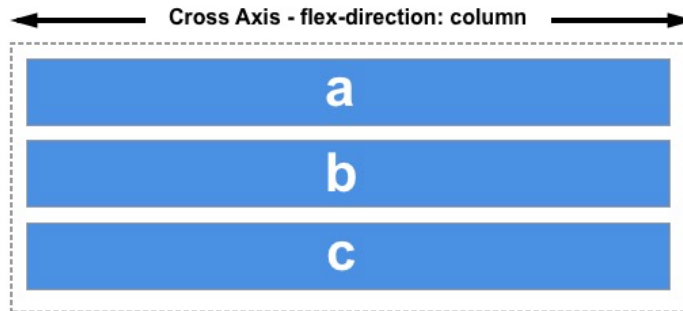
# CSS FLEXBOX

- 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

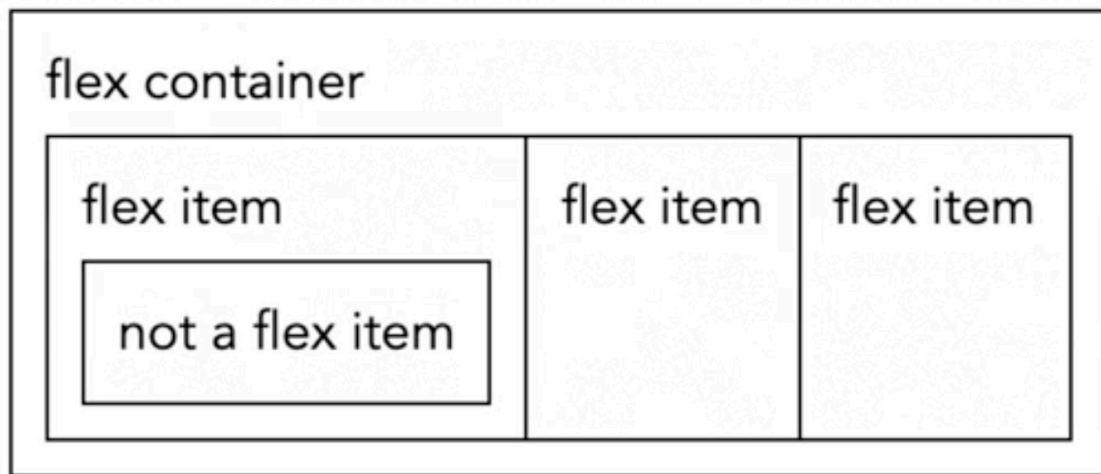
# ROW AXIS



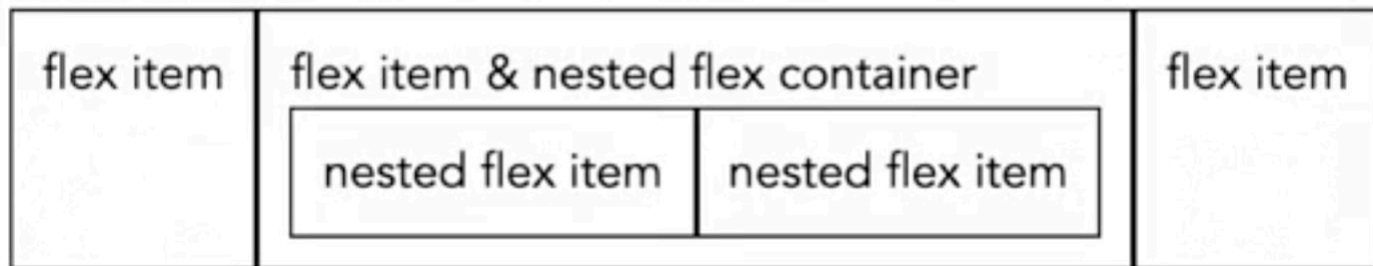
# COLUMN AXIS







flex container



# FLEXBOX PROPERTIES

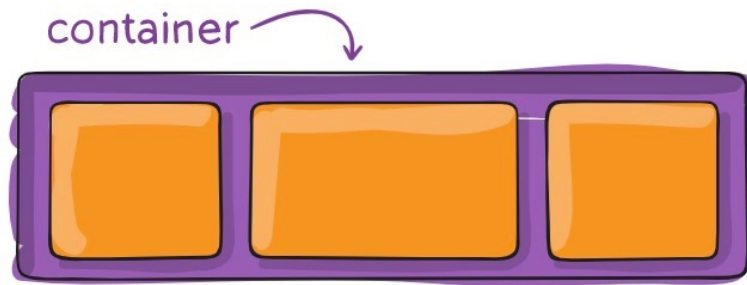
## FOR THE PARENT

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

## FOR CHILDREN

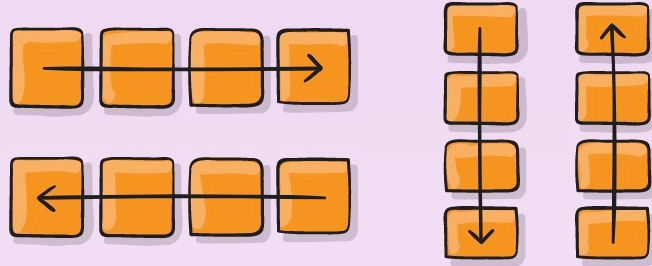
- flex-grow
- flex-shrink
- flex-basis
- **flex**
- align-self
- order

# PROPERTIES FOR THE PARENT (flex container)



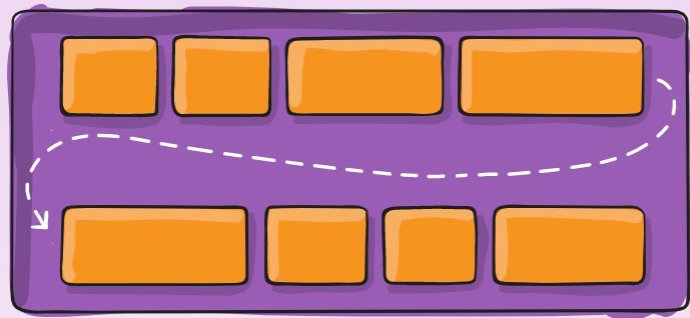
```
.container {  
  display: flex; /* or inline-flex */  
}
```

## 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



```
.container {  
  flex-flow: column wrap;  
}
```

# justify-content

flex-start



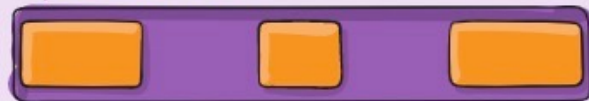
flex-end



center



space-between



space-around



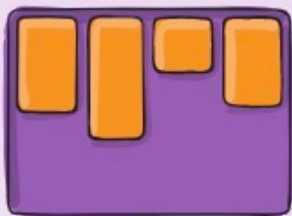
space-evenly



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

## align-items

flex-start



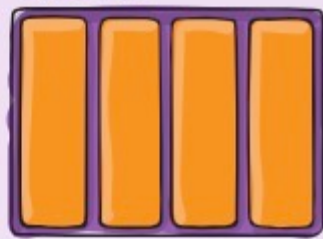
flex-end



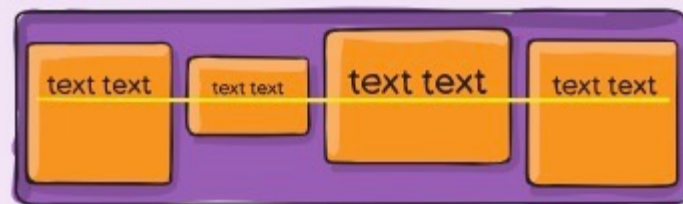
center



stretch



baseline

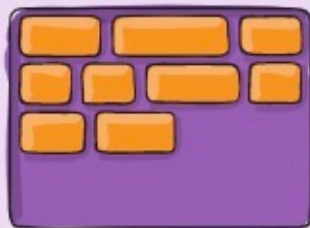


```
.container {  
  align-items: flex-start | flex-end | center |  
stretch | baseline;  
}
```



## align-content

flex-start



flex-end



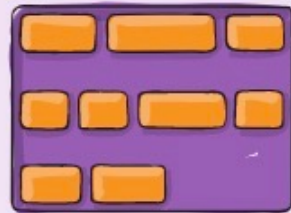
center



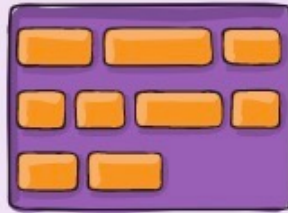
stretch



space-between



space-around



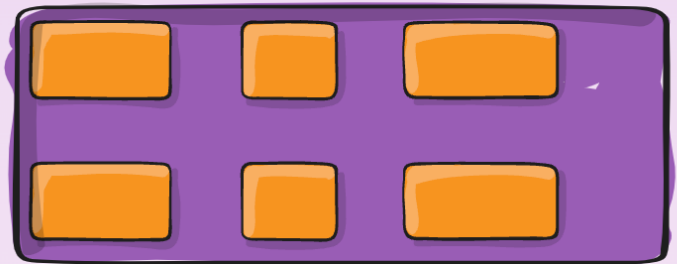
```
.container {  
  align-items: flex-start | flex-end | center |  
  stretch | space-between | space-around;  
}
```

## gap, row-gap, column-gap

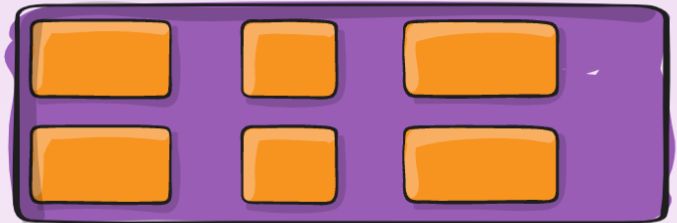
gap: 10px



gap: 30px



gap: 10px 30px

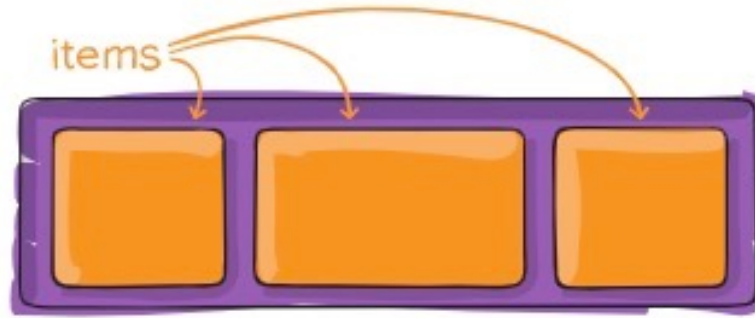


```
.container {  
  display: flex;  
  ...  
  gap: 10px;  
  gap: 10px 20px; /* row-gap column gap */  
  row-gap: 10px;  
  column-gap: 20px;  
}
```

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

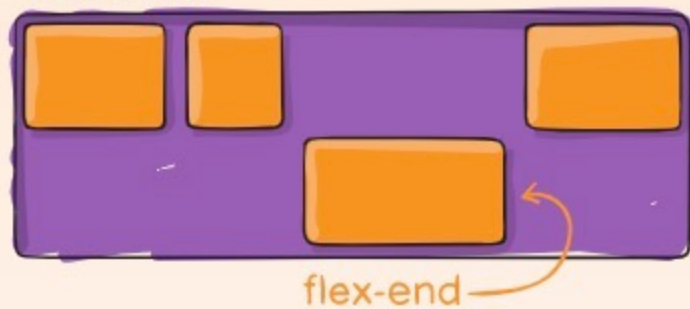
This is the shorthand for `flex-grow`, `flex-shrink` and `flex-basis` combined. The second and third parameters ( `flex-shrink` and `flex-basis` ) are optional. The default is `0 1 auto` , but if you set it with a single number value, it's like `1 0` .



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

## align-self

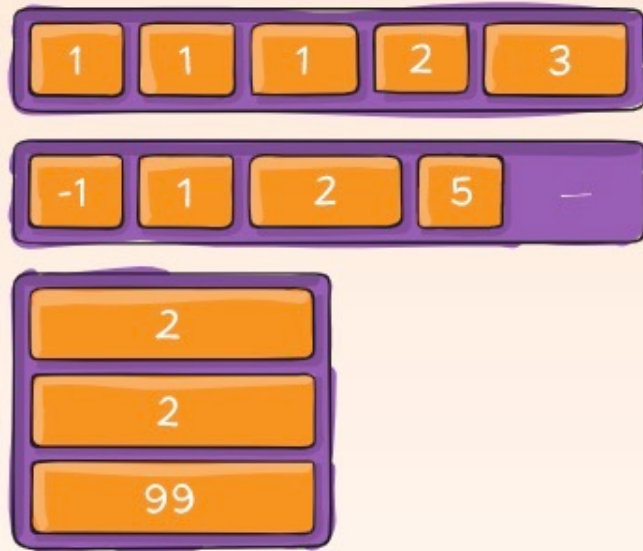
flex-start



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



## order



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

## justify-content

flex-start



space-around



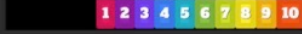
center



space-between



flex-end



space-evenly



## align-content

flex-start



center



flex-end



space-around



space-between



space-evenly



stretch



## align-items

stretch



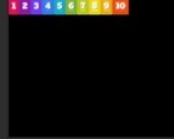
stretch



baseline



flex-around



center



flex-end



## align-self

flex-start



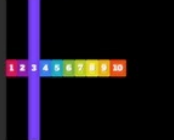
center



flex-end



stretch



baseline



auto

