# Flexbox

[TOC]

## CSS3 Flexbox Concepts

Flexbox consists of flex (=flexible) containers and flex items.

**main axis (=eje)** - The main axis of a flex container is the primary axis along which flex items are laid out (=desplegado, puesto). Beware (=tener cuidado), it is not necessarily horizontal; it depends on the flex-direction property (see below (=abajo)).

**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 (=cualquiera que sea) 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 (=estar relleno de) 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.

## Container Properties

A flex container is declared by setting the display property of an element to either flex(rendered (=reproducido) as a block) or inline-flex (rendered as inline).

Inside a flex container there is one or more flex items.

**Note:** Everything outside a flex container and inside a flex item is rendered as usual. Flexbox defines how flex items are laid out (=desplegado, puesto) inside a flex container.

Flex items are positioned inside a flex container along a flex line. By default there is only one flex line per flex container.

The following example shows three flex items. They are positioned by default: along the horizontal flex line, from left to right:

in index.html:

*<!-- index.html -->*

<div class="container">

<div class="item">1</div>

<div class="item">2</div>

<div class="item">3</div>

<div class="item">4</div>

<div class="item">5</div>

<div class="item">6</div>

<div class="item">7</div>

<div class="item">8</div>

</div>

in main.css:

*/\* main.css \*/*

.container {

border: 1px solid #0A77E9;

display: flex;

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

}

.item{

color: white;

font-family: sans-serif;

background: #0A77E9;

padding: 5px 15px;

margin: 5px;

border-radius: 6px;

}

[**Example codepen**](http://codepen.io/zapatran/pen/VjoXOY) (<http://codepen.io/zapatran/pen/VjoXOY> )

### Flex Direction

flex-direction property specifies the direction of the items inside the container.

The default value of flex-direction is row (left-to-right, top-to-bottom). In the dafault value all items will try to fit all the way up (=alto)

row-reverse - If the writing-mode (direction) is left to right, the flex items will be laid out right to left column - If the writing system is horizontal, the flex items will be laid out verticallycolumn-reverse - Same as column, but reversed

.container {

...

display: flex;

flex-direction: row;

*/\* flex-direction: row-reverse; \*/*

*/\* flex-direction: column; \*/*

*/\* flex-direction: column-reverse; \*/*

}

...

[**Example codepen**](http://codepen.io/zapatran/pen/VjoXOY) (<http://codepen.io/zapatran/pen/VjoXOY>)

### Flex Wrap (=envolver, rodear)

flex-wrap specifies if items should wrap or not inside the container if the is enough space for them on one flex line.

The possible values are:

nowrap - Default value. The flexible items will not wrap

wrap - The flexible items will wrap if necessary

wrap-reverse - The flexible items will wrap, if necessary, in reverse order.

.container {

border: 1px solid #0A77E9;

display: flex;

flex-direction: row;

flex-wrap: wrap;

*/\* flex-wrap: nowrap; \*/*

*/\* flex-wrap: wrap-reverse; \*/*

}

.item{

color: white;

font-family: sans-serif;

background: #0A77E9;

padding: 5px 15px;

margin: 5px;

border-radius: 6px;

width: 150px; */\* added \*/*

}

[**Example codepen**](http://codepen.io/zapatran/pen/rWeJOq) (<http://codepen.io/zapatran/pen/rWeJOq> )

### Flex justify-content

justify-content aligns all the items alongs all the available space.

The possible values are:

flex-**start** - Default value. Items are positioned at the beginning of the container

flex-**end** - Items are positioned at the end of the container

center - Items are positioned at the center of the container

space-between - Items are positioned with space between the lines

space-around - Items are positioned with space before, between, and after the lines

.container {

border: 1px solid #0A77E9;

display: flex;

flex-direction: row;

flex-wrap: wrap;

*/\* flex-wrap: nowrap; \*/*

*/\* flex-wrap: wrap-reverse; \*/*

*/\* justify-content: flex-start; \*/*

*/\* justify-content: flex-end; \*/*

justify-content: center;

*/\* justify-content: space-between; \*/*

*/\* justify-content: space-around; \*/*

}

.item{

color: white;

font-family: sans-serif;

background: #0A77E9;

padding: 5px 15px;

margin: 5px;

border-radius: 6px;

width: 150px;

}

[**Example Codepen**](http://codepen.io/zapatran/pen/JbXpKQ) (<http://codepen.io/zapatran/pen/JbXpKQ> )

### Flex align-items

The align-items property vertically aligns the flexible container's items when the items do not use all available space on the cross-axis.

The possible values are:

stretch - Default value. Items are stretched to fit the container

flex-**start** - Items are positioned at the top of the container

flex-**end** - Items are positioned at the bottom of the container

center - Items are positioned at the center of the container (vertically)

baseline - Items are positioned at the baseline of the container

<div class="container">

<div class="item">1</div>

<div class="item">2</div>

<div class="item">3</div>

<div class="item">4</div>

<div class="item height1">5</div>

<div class="item">6</div>

<div class="item height1">7</div>

<div class="item">8</div>

</div>

.container {

border: 1px solid #0A77E9;

display: flex;

height: 500px;

*/\* flex-direction: row; \*/*

*/\* flex-wrap: wrap; \*/*

*/\* flex-wrap: nowrap; \*/*

*/\* flex-wrap: wrap-reverse; \*/*

*/\* justify-content: flex-start; \*/*

*/\* justify-content: flex-end; \*/*

*/\* justify-content: center; \*/*

*/\* justify-content: space-between; \*/*

*/\* justify-content: space-around; \*/*

*/\* align-items: stretch; \*/*

*/\* align-items: flex-start; \*/*

*/\* align-items: flex-end; \*/*

*/\* align-items: center; \*/*

align-items: baseline;

}

.item{

color: white;

font-family: sans-serif;

background: #0A77E9;

padding: 5px 15px;

margin: 5px;

border-radius: 6px;

}

.height1 {

height: 200px;

font-size: 3em;

display: flex;

align-items: center;

}

We added .height1 to see the behavior of align-items: baseline in .container class.

[**Example Codepen**](http://codepen.io/zapatran/pen/ObNQjg) (<http://codepen.io/zapatran/pen/ObNQjg> )

### Flex align-content

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

The possible values are:

stretch - Default value. Lines stretch to take up the remaining space

flex-**start** - Lines are packed toward the start of the flex container

flex-**end** - Lines are packed toward the end of the flex container

center - Lines are packed toward the center of the flex container

space-between - Lines are evenly distributed in the flex container

space-around - Lines are evenly distributed in the flex container, with half-size spaces on either end

<div class="container">

<div class="item">1</div>

<div class="item">2</div>

<div class="item">3</div>

<div class="item">4</div>

<div class="item">5</div>

<div class="item">6</div>

<div class="item">7</div>

<div class="item">8</div>

</div>

.container {

border: 1px solid #0A77E9;

display: flex;

height: 500px;

flex-direction: row;

flex-wrap: wrap;

*/\* flex-wrap: nowrap; \*/*

*/\* flex-wrap: wrap-reverse; \*/*

*/\* justify-content: flex-start; \*/*

*/\* justify-content: flex-end; \*/*

*/\* justify-content: center; \*/*

*/\* justify-content: space-between; \*/*

*/\* justify-content: space-around; \*/*

*/\* align-items: stretch; \*/*

*/\* align-items: flex-start; \*/*

*/\* align-items: flex-end; \*/*

*/\* align-items: center; \*/*

*/\* align-items: baseline; \*/*

*/\* align-content: stretch; \*/*

*/\* align-content: flex-start; \*/*

*/\* align-content: flex-end; \*/*

align-content: center;

*/\* align-content: space-around; \*/*

*/\* align-content: space-between; \*/*

}

.item{

color: white;

font-family: sans-serif;

background: #0A77E9;

padding: 5px 15px;

margin: 5px;

border-radius: 6px;

width: 150px;

}

[**Example Codepen**](http://codepen.io/zapatran/pen/ZBWrvw) (<http://codepen.io/zapatran/pen/ZBWrvw> )

## Items Properties

### Order

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

<div class="container">

<div class="item">1</div>

<div class="item">2</div>

<div class="item">3</div>

<div class="item">4</div>

<div class="item order1">5</div>

<div class="item">6</div>

<div class="item">7</div>

<div class="item">8</div>

</div>

.container {

border: 1px solid #0A77E9;

display: flex;

flex-direction: row;

}

.item{

color: white;

font-family: sans-serif;

background: #0A77E9;

padding: 5px 15px;

margin: 5px;

border-radius: 6px;

}

.order1 {

order: 1;

}

[**Example Codepen**](http://codepen.io/zapatran/pen/QGNQxB) (<http://codepen.io/zapatran/pen/QGNQxB> )

### flex-grow

The flex property specifies the length of the flex item, relative to the rest of the flex items inside the same container.

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, the remaining space would take up twice as much space as the others (or it will try to, at least).

<div class="container">

<div class="item item1">1</div>

<div class="item item2">2</div>

</div>

.container {

border: 1px solid #0A77E9;

display: flex;

flex-direction: row;

width: 1000px;

margin: 0 auto;

}

.item1{

background-color: red;

}

.item2{

background-color: #0A77E9;

}

.item{

color: white;

font-family: sans-serif;

padding: 5px;

width: 400px;

box-sizing: border-box;

flex-grow: 1;

}

[**Example Codepen**](http://codepen.io/zapatran/pen/KNzQYb) (<http://codepen.io/zapatran/pen/KNzQYb> )

In the example above, we have a container with 1000px width and two items inside with400px width each one. So the free space is 200px.

If we define the flex-grow: 1; property for every item, they will distribute the free space between them. item1 and item2 will have 500px width each.

Another example, Imagine that item1 has flex-grow: 2; and item2 has flex-grow: 1;, the free space is 200px and we have to distribute that space by 3.

.item1{

background-color: red;

flex-grow: 2;

}

.item2{

background-color: #0A77E9;

flex-grow: 1;

}

.item{

...

*/\* flex-grow: 1; \*/*

}

200px/3 = 66,66666px

item1 will have 400px + 66,6666px\*2 = 533px and item2 will have400px + 66,6666px = 467px

### flex-shrink

This defines the ability for a flex item to shrink if necessary. Basically is the same concept thatflex-grow but when **flex items doesn't fit** in the **container**.

.container {

border: 1px solid #0A77E9;

display: flex;

flex-direction: row;

width: 600px;

margin: 0 auto;

}

.item1{

background-color: red;

flex-shrink: 2;

}

.item2{

background-color: #0A77E9;

flex-shrink: 1;

}

.item{

color: white;

font-family: sans-serif;

padding: 5px;

width: 400px;

box-sizing: border-box;

flex-grow: 1;

}

[**Example Codepen**](http://codepen.io/zapatran/pen/LbZYXz) (<http://codepen.io/zapatran/pen/LbZYXz> )

### flex-basis

This defines the default size (**width** or **height** depending on the main axis flex-direction) of an element before the remaining space is distributed. It can be a length (e.g. 20%, 5rem, etc.) or a keyword.

.container {

border: 1px solid #0A77E9;

display: flex;

flex-direction: row;

width: 1000px;

margin: 0 auto;

}

.item1{

background-color: red;

flex-basis: 400px;

}

.item2{

background-color: #0A77E9;

flex-basis: 400px;

}

.item{

color: white;

font-family: sans-serif;

padding: 5px;

width: 400px;

box-sizing: border-box;

}

[**Example Codepen**](http://codepen.io/zapatran/pen/mOEeeB) (<http://codepen.io/zapatran/pen/mOEeeB> )

### align-self

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

See the align-items explanation to understand the available values.

<div class="container">

<div class="item">1</div>

<div class="item">2</div>

<div class="item center">3</div>

<div class="item">4</div>

<div class="item">5</div>

<div class="item end">6</div>

<div class="item">7</div>

<div class="item">8</div>

</div>

.container {

border: 1px solid #0A77E9;

display: flex;

flex-direction: row;

height: 500px;

}

.item{

color: white;

background-color: #0A77E9;

font-family: sans-serif;

padding: 5px;

margin: 5px;

height: 50px;

box-sizing: border-box;

flex-grow: 1;

}

.end {

*/\* align-self: stretch; \*/*

*/\* align-self: flex-start; \*/*

align-self: flex-end;

*/\* align-items: center; \*/*

*/\* align-items: baseline; \*/*

}

.center {

align-self: center;

}

[**Example Codepen**](http://codepen.io/zapatran/pen/yVJYde) (<http://codepen.io/zapatran/pen/yVJYde> )





