Web design and develpment II (CW2)

Class 2

Columns layout

Cum sociis natoque penatibus et magnis dis parturient sagittis lacus vel augue laoreet.

Etiam habebis sem dicantur magna mollis euismod. Integer legentibus erat a ante historiarum dapibus. Donec sed odio operae, eu vulputate felis rhoncus. Quisque placerat facilisis egestas cillum dolore. Quo usque tandem abutere, Catilina, patientia nostra? Qui ipsorum lingua Celtae, nostra Galli appellantur. Tu quoque, Brute, fili mi, nihil timor populi, nihil! Cum ceteris in veneratione tui montes, nascetur mus.

Vivamus sagittis lacus vel augue laoreet rutrum faucibus. Qui ipsorum lingua Celtae, nostra Galli appellantur. Praeterea iter est quasdam res quas ex communi. Integer legentibus erat a ante historiarum dapibus.

Petierunt uti sibi concilium totius Galliae in diem certam indicere. Integer legentibus erat a ante historiarum dapibus. Quid securi etiam tamquam eu fugiat nulla pariatur. Unam incolunt Belgae, aliam Aquitani, tertiam. Fictum, deserunt mollit anim laborum astutumque! Etiam habebis sem dicantur magna mollis euismod.

Just like in a newspaper, it is possible to layout web contents using columns.

Defining columns

To define a container as columns, « column-count » property and indicate the number of columns needed.

Gutter size

The space between columns is called « gutter », and its size can be specified using the property « column-gap ».

Styling rules

Vertical lines used as separators are called « rules », and it is possible to ajust them using the following properties : « column-rule-style », « column-rule-width » and « column-rule-color ».

Rules shorthand :
column-rule: 1px solid lightblue;

Columns span

This property allows an element to span over a specific number of columns (like the title in the example at the top of this page).

CSS :

```
.newspaper {
        column-count: 3;
        column-gap: 40px;
        column-rule-style: solid;
        column-rule-width: 1px;
        column-rule-color: lightblue;
}

h2 {
        column-span: all;
}
```

HTML :

```
<article class="newspaper">

<h2>Lorem Ipsum Dolor Sit Amet</h2>

Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobo...

</article>
```

Flex

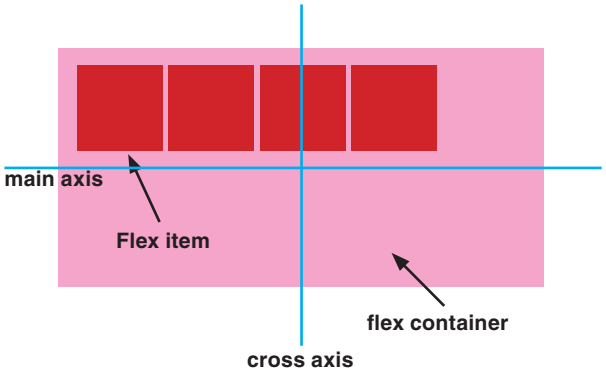
Flex, as its name indicates, brings even more flexibility into the task of positioning elements. It also makes producing responsive web pages even easier, and offering more possibilities than other approaches.

Defining a flex container (flexbox)

The first step in flex positioning is to define a container as flex. It is possible to nest non flex elements into a flexbox or for a flex-box to be nested into a non flex container.-

To define a container as flex : **display: flex;**

All the element of a flex container are considered flex items.



Display items in row or column

The flex container parameters indicates whether to display the items vertically (**column**) or horizontally (**row**) using the property **flex-direction**.

Horizontal alignment

The flex container parameters also indicates how to align its items horizontally using the property **justify-content** which possible values are : *flex-start (left), flex-end (right), center, space-around* and *space-between*.

Vertical alignment

To align items vertically in a flex-box, the property align-items is used along with its possible values : *stretch, center, flex-start (top), flex-end (bottom)* and *baseline*.

CSS :

```
.container {
        display: flex;
        flex-direction: row;
        justify-content: flex-start;
        align-items: flex-start;
        width: 500px;
        height: 200px;
        bacground-color: pink;
}

.item {
        width: 50px;
        height: 50px;
        bacground-color: red;
}
```

HTML :

```
<div class="container">
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
        <div class="item"> </div>
</div>
```

Horizontal distribution of items

Two values of the property **justify-content** are very useful: **space-between** and **space-around.**

The value **space-between** position the first and the last items at both extremes, and the other in-between elements equally distant.

The value **space-around** creates equal left and right margins on each of the items.

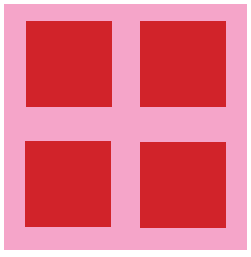
Wrapping items

In order to produce responsive web pages, flex helps managing interfaces making it possible to make them so they adapt to different browsers window or viewport sizes.

Wrapping items makes it possible to indicates to items to wrap, meaning that when the container becomes too narrow, instead of shrinking, items display on as many lines as necessary.

To do so, the property **flex-wrap** is used along with the values **wrap** or **nowrap**.

wrap-reverse may also be used so the items are displayed in inverted order.



```
.container {
        display: flex;
        flex-direction: row;
        justify-content: space-between;
}
```

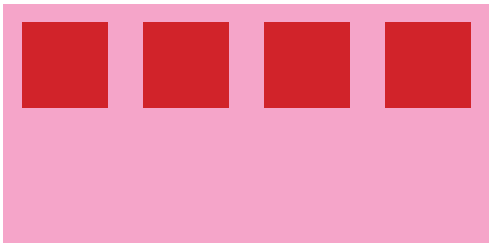
```
.container {
        display: flex;
        flex-direction: row;
        justify-content: space-around;
}
```

```
.
container {
        display: flex;
        flex-direction: row;
        justify-content: space-around;
        flex-wrap: wrap;
}
```

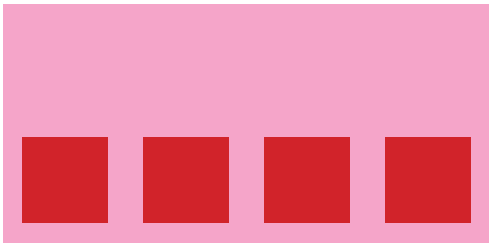
Vertical distribution of items

As we've already seen, the property **align-items** is used for vertical alignment.

flex-start

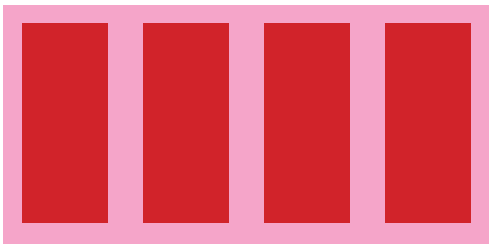


flex-end



By default, the value stretch is set which mean that if an items has no height defined, it will stretch vertically to the container's height.

stretch



Note that properties and values **float**, **clear** and **vertical-align** cannot be used with flex and that they basically have no reason of being anymore.

```
.flexcontainer {
        display: flex;
        flex-direction: row;
        justify-content: space-around;
        flex-wrap: wrap;
        align-items: flex-start;
}
```

```
.flexcontainer {
        display: flex;
        flex-direction: row;
        justify-content: space-around;
        flex-wrap: wrap;
        align-items: flex-end;
}
```

```
.flexcontainer {
        display: flex;
        flex-direction: row;
        justify-content: space-around;
        align-items: stretch;
}
```

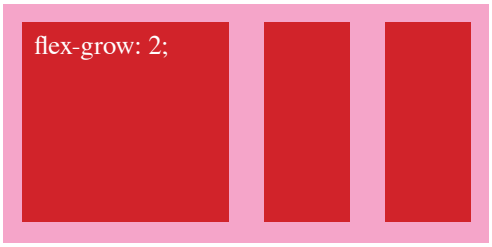
Items dimensions

Flex-grow

In order for each item of a container to use an equal space, **flex-grow: 1** may be used.

If there is 3 items, each marked flex-grow: 1, this means each will used 1 unit out of 3 (1/3).

If there is 3 items, but we would like for the first item to use more space than the other, we can indicate flex-grow: 2 to this item. This means the fist item will use 2 units of space out of 4 (2/4 or 1/2).



While flex-grow indicate how an item can grow, another property, **flex-shrink**, indicates how much it can shrink.

```
.flexcontainer {
        display: flex;
        flex-direction: row;
}

.itemA {
        flex-grow: 2;
}

.itemB,
.itemC {
        flex-grow: 1;
}
```

flex-basis

Flex-basis indicates the space items should accupiy depending on its value:

auto:
Equally distributes remaining space based of items volume of content.

0:
Organizes space sizing the items with no regards to their content's volume.

Absolute or relative measurement:
Precise absolute (px, mm, etc.) or relative (%, etc.) measurements are used.