**CSS flex**

* The Flexible Box Layout Module makes it easier to design flexible responsive layout structure without using float or positioning.
* A Flexible Layout must have a parent element with the display property set to flex. Direct child elements(s) of the flexible container automatically becomes flexible items.

.flex-container {

display: flex;

}

<div class="flex-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

* The flex container properties are:
  + flex-direction
  + flex-wrap
  + flex-flow
  + justify-content
  + align-items
  + align-content
* **justify-content**
  + This property aligns items horizontally and accepts the following values-
    - **flex-start**: Items align to the left side of the container.
    - **flex-end**: Items align to the right side of the container.
    - **center**: Items align at the center of the container.
    - **space-between**: Items display with equal spacing between them.
    - **space-around**: Items display with equal spacing around them.
* **align-items**
  + This property aligns items vertically and accepts the following values-
    - **flex-start**: Items align to the top of the container.
    - **flex-end**: Items align to the bottom of the container.
    - **center**: Items align at the vertical center of the container.
    - **baseline** : Items display at the baseline of the container.
    - **stretch**: Items are stretched to fit the container.
* **flex-direction**
  + This property defines the direction items are placed in the container -
    - **row**: Items are placed the same as the text direction.
    - **row-reverse**: Items are placed opposite to the text direction.
    - **column**: Items are placed top to bottom
    - **column-reverse** : Items are placed bottom to top.
  + **Note** that when you set the direction to a reversed row or column, start and end are also reversed i.e. justify-content:flex-end will mean flex-start……
  + **Note** that when the flex direction is a column, justify-content changes to the vertical and align-items to the horizontal.
* **order**
  + Sometimes reversing the row or column order of a container is not enough. In these cases, we can apply the order property to individual items.
  + By default, items have a value of 0, but we can use this property to also set it to a positive or negative integer value (-2, -1, 0, 1, 2).
  + This property has to be applied to a child component (individual item).
* **align-self**
  + Another property you can apply to individual items is align-self. This property accepts the same values as align-items and its value for the specific item (child component).
* **flex-wrap**
  + The flex-wrap property specifies whether the flex items should wrap (multiple line) or not.
  + This property defines the direction items are placed in the container -
    - **wrap**: Items wrap around to additional lines.
    - **nowrap**: Every item is fit to a single line.
    - **wrap-reverse**: Items wrap around to additional lines.
* **flex-flow**
  + The two properties flex-direction and flex-wrap are used so often together that the shorthand property flex-flow was created to combine them. This shorthand property accepts the value of one of the two properties separated by a space.
  + flex-flow: flex-direction flex-wrap.
* **align-content**
  + This property is used to set how multiple lines are spaced apart from each other-.
    - **flex-start**: Lines are packed at the top of the container.
    - **flex-end**: Lines are packed at the bottom of the container.
    - **center**: Lines are packed at the vertical center of the container.
    - **space-between** : Lines display with equal spacing between them.
    - **space-around**: Lines display with equal spacing around them.
    - **stretch**: Lines are stretched to fit the container.
  + This can be confusing, but align-content determines the spacing between lines, while align-items determines how the items as a whole are aligned within the container. When there is only one line, align-content has no effect.