CSS LINK

<link rel="stylesheet" type="text/css" href="style.css">

JS HTML DOM addEventListener()

-The addEventListener() method attaches an event handler to the specified element. تقوم بإرفاق معالج أحداث بالعنصر المحدد.

- *element*.addEventListener(*event*, *function*, *useCapture*)

- event : Required. A String that specifies the name of the event.

events: https://www.w3schools.com/jsref/dom_obj_event.asp

- function: Required. Specifies the function to run when the event occurs.

- `document.getElementById("id").addEventListener("click", function(){});`

- **Tip:** Use the removeEventListener() method to remove an event handler that has been attached with the addEventListener() method.

-Tip: Use the document.addEventListener() method to attach an event handler to the document.

Ex:

```
<!DOCTYPE html>
<html>
<body>

<p>This example uses the addEventListener() method to attach a click event to a
button.</p>

<button id="myBtn">Try it</button>

<p id="demo"></p>

<script>
document.getElementById("myBtn").addEventListener("click", function(){
    document.getElementById("demo").innerHTML = "Hello World";
});
</script>

</body>
</html>
```

In Browser

This example uses the addEventListener() method to attach a click event to a button.

Try it

Hello World

CSS The id Selector

-To select an element with a specific id, write a hash (#) character, followed by the id of the element.

-**Note:** An id name cannot start with a number!

Ex:

```
<body>
<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>
</body>

#para1 {
   text-align: center;
   color: red;
}
```

Css The class Selector

-To select elements with a specific class, write a period (.) character, followed by the name of the class.

Ex:

```
<body>
<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>
</body>

.center {
   text-align: center;
   color: red;
}
```

-You can also specify that only specific HTML elements should be affected by a class.

In the example below, only <p> elements with class="center" will be center-aligned:

Ex:

```
<body>
<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
</body>
```

```
p.center {
   text-align: center;
   color: red;
}
```

-HTML elements can also refer to more than one class.

In the example below, the <p> element will be styled according to class="center" and to class="large":

Ex:

```
<body>

<h1 class="center">This heading will not be affected</h1>

<p class="center">This paragraph will be red and center-aligned.</p>

<p class="center large">This paragraph will be red, center-aligned, and in a large font-size.</p>

</body>
```

```
p.center {

   text-align: center;

   color: red;

}

p.large {

   font-size: 300%;

}
```

-If you have elements with the same style definitions, like this:

```
h1, h2, p {
    text-align: center;
    color: red;
}
```

HTML CSS ../

```
<link rel="stylesheet" href="../styles/firstday.css"
```

../ it means go to before folder

Html Default Block element

Content 100%(width), padding 0% (0% top + 0% bottom 0% left + 0% right), border 0%(0% top + 0% bottom 0% left + 0% right) are as default for every block element.

Ex:

```
h1{

    width: 50%;
    padding: 0% 25%;
    border: 0px;
}
```

50+25+25 =100

Ex2:

Screen 1000px

Content+padding+border

```
h1{

    width: 50%;
    padding: 0% 25%;
    border: 1px solid black;
}
```

What is h1 size from screen ?

Width: 50% = 50% from 1000 px= 500px

padding: 0% 25% = 250px left +250px right=500

border: 1px left +1px left right = 2px

so h1 size = 500+500+2= 1002px ohhhh that is problem

```
h1{

    width: 50%;
    padding: 0% 10%;
    border: 1px solid black;
    box-sizing: border-box;
    margin: 0% auto;
}
```

Css never fixed width and height

Never define fixed width and height

css units

```
    <div>
```

```
        <a href="#">Dortmund</a>
        <a href="#">Bayern</a>

    </div>
```

div 60%

a 10% : 10% from div width

a 10vw : 10% frim device width

a 10px : will be always 10px, and it is not fixable so that not god

css Block vs Inline

```
text-align: center;
```

that will effect on inline elements and inside of Block elements, not on Block elements.

So the block elements will be not in center

Css Media queries

```
@media screen and (max-width: 650px){ /* for Tablate. if width up to 650px ,
then  do the function. else do the orginal  */
    nav > a {
        font-size: 4vw;
        background: red;
    }
}
```

```
@media screen and (max-width:400px){ /* for Mobile. if width up to 400px ,
then  do the function. else do the orginal  */
    nav > a {
        display: block;
    }
}
```

Css uniq selector

```
P:nth-of-chils(2) {}
```

That mean if p is the 2th child then do

Css flexbox

```
section{
    display: flex;
    flex-direction: column;
    justify-content: space-between;
}
```

Here the section is flex-container

```
    <section>
        <div></div>
        <div></div>
        <div></div>
    </section>
```

Here the divs are flex-iteams

```
div:nth-of-type(1) {
    flex-grow: 1;
    order: 2;
}
div:nth-of-type(2) {
    flex-grow: 2;
    order: 1;
}
div:nth-of-type(3) {
    flex-grow: 0;
    order: 3;
}
```

Div 1 will take 1-unit from rest space + order 2

Div 2 will take 2-unit from rest space + order 2

Div 3 will take 0-unit from rest space + order 3

Css flex-basis

```
div:nth-of-type(2) {
    flex-basis: 100px;
}
```

The `flex-basis` property specifies the initial length of a flexible item.

Note:

```
section > div:nth-last-of-type(1) {
    background-image: url(./images/goettingen.jpg);
}
section > div:nth-last-of-type(2) {
    background-image: url(./images/lueneburg.jpg);
}
section > div:nth-last-of-type(3) {
    background-image: url(./images/trier.jpg);
}
```

```
section > div {

    flex-basis: 20%;
    transition: flex-basis 1s linear;
}

section > div:hover {
```

```
    flex-basis: 30%;
}
```

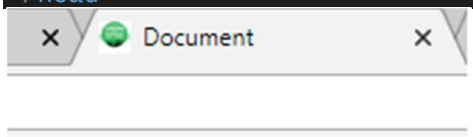
Transation for flex-basis not working with img…

it works with div… so the solution is to set the image as background for div =➔ background-image: url(./images/goettingen.jpg);

css Icon link

https://www.favicon-generator.org/ …. To generate 16*16 image size or icon

```
<head>
    <link rel="icon" href="../images/img123.ico">
</head>
```



css Flex-container x-axis vs y-axis

```
Flex-container
section {
    display: flex; it will make the section as flex-container
    flex-direction: row;   the iteams inside this container will go on as row
    justify-content: center; items will be in the center of X-axis
    align-items: center; items will be in the center of y-axis
```

but

```
flex-direction: row;   .... x-axis is the row, y-axis is the Colom
flex-direction: Colom;   .... x-axis is the Colom, y-axis is the row
```

css Flex-shrink

The `flex-shrink` property specifies how the item will shrink relative to the rest of the flexible items inside the same container. Shrink=انكماش

So with flex-shrink (no overflow).

Css 14_flex_project

```
* {
  }
```

That means all elements

……

to hidden elements with transition (never use display:none; nor visibility:hidden;)

opacity غموض: 0; + transition

or

height: 0; + transition

width: 0; + transition

.....

Html 14_flex_project

```
a[href="#"]*5+li>img
        <a href="#"></a>
        <a href="#"></a>
        <a href="#"></a>
        <a href="#"></a>
        <a href="#"></a>
        <li><img src="" alt=""></li>
```

.....

```
button:hover ~ article {
    make hover to all article elements of(button's siblings)
}

button:hover {
    make hover to this button
}

button:focus + p {
    when I press, do the action to only next  p after button
}

button:focus ~ p {
    when I press, do the action to all p elements(button's siblings)
}

button:focus {
    when I press, do the action to this button
}
```

HTML description list

```
    <h1>World cup group phase</h1>
    <dl>
        <dt>Grpup A teams</dt>
            <dd>Brazil</dd>
```

```
            <dd>Russia</dd>
            <dd>Germany</dd>

        <dt>Grpup B teams</dt>
            <dd>France</dd>
            <dd>Spain</dd>
            <dd>Iran</dd>

    </dl>
```

World cup group phase

Grpup A teams
    Brazil
    Russia
    Germany
Grpup B teams
    France
    Spain
    Iran

Grid Layout

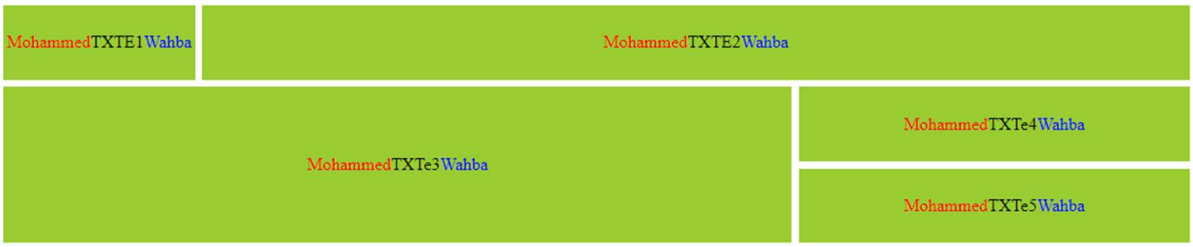
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

```
.grid-container {
    width: 80vw;

    margin: 5vw auto;
    display: grid;
    grid-template: repeat(8,5vw) / repeat(6, 1fr); .. (row / columm)only width
    grid-gap: 0.5vw 0.5vw;


}
.grid-items {
 grid-area: 1 / 2 / span 1 / span 5;..row / columm /امتداد  span row /span columm

}
```

css pseudo-selectors

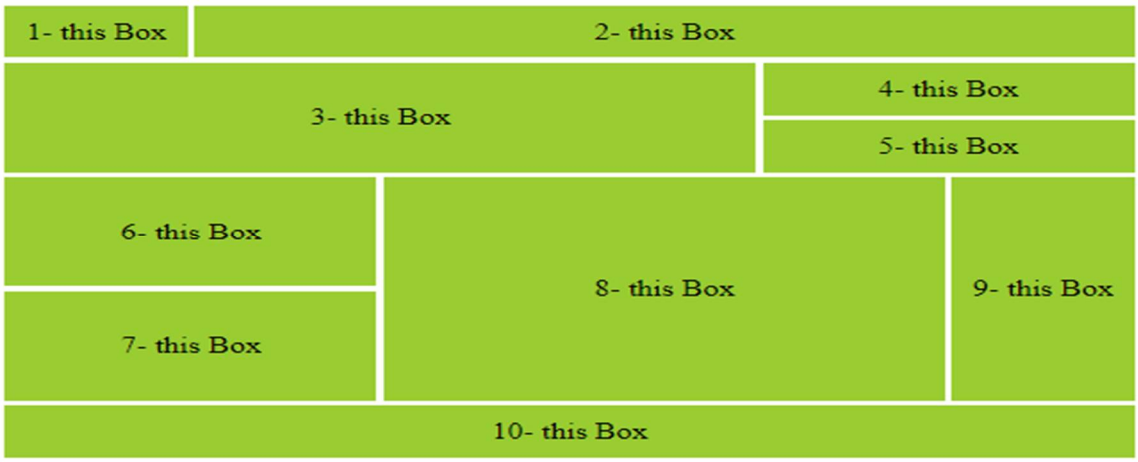
it can not to copy and mark it

```
div p:before {
    content: 'Mohammed';
    color:red;
}
div p:after {
    content: 'Wahba';
    color:blue;
}
```



Css conuter

```
:root {
    counter-reset: variabale;   /*  variable= 0 */
}
div p:before {
     counter-increment: variabale;    /*  variable= variable + 1 */
    content: counter(variabale)  '-  this Box';  /*  counter(variabale) that
mean print variable */
}
```



Css unlist + conuter

```
        <div class="it8 grid-items">
            <h2>Drinks</h2>

                <ul>
                <li>Coffee</li>
                <li>Tea</li>
                <li>Milk</li>
                </ul>
        </div>
```

```
:root {
    counter-reset: variabale;   /*  variable= 0 */
}

.it8 li:before {
    counter-increment: variabale;    /*  variable= variable + 1 */
```

```
   content: counter(variabale) '. ';  /*  counter(variabale) that mean print
variable */
}
ul {
    list-style-type: none;     to remove the big dot from ul
}
```

Css public variables

Css Variables, we define them either in :root or body.

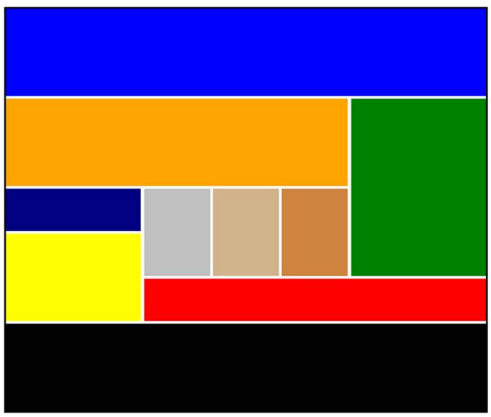
They are case sensitive , var(--myVarible).

```
:root {
    counter-reset: variabale;   /*  variable= 0 */
    --varible-myBlue: #32E1FF;
    --big-font: 3.5vw;
    --small-font: 1.5vw;
    --image-adress: https://www.gooodkdmdn,
}
.it8 {
    background-color: var(--varible-myBlue)
}
```

HTML CSS Grafic Grid

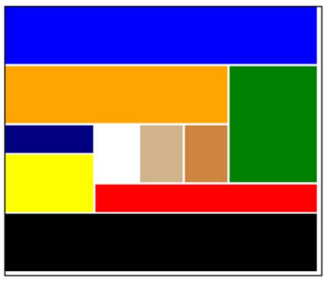
```
.grid-container {
    width: 80vw;
    margin: 5vw auto;
    border: 2px solid black;
    display: grid;
    grid-template: repeat(9, 7vw) / repeat(7, 1fr);
    grid-template-areas:   'b b b b b b b'
                           'b b b b b b b'
                           'o o o o o g g'
                           'o o o o o g g'
                           'c c s t u g g'
                           'y y s t u g g'
```

```
                            'y y r r r r r'
                            'k k k k k k k'
                            'k k k k k k k';
    grid-column-gap: 0.5vw;
    grid-row-gap: 0.5vw;


}

.it1 {
    background: blue;
    grid-area: b;

}
.it2 {

    background:orange;
    grid-area: o;

 }
.it3 {

    background: green;
    grid-area: g;

}
.it4 {

    background: silver;
    grid-area: s;

}
.it5 {

    background:tan;
    grid-area: t;

}
```



```
    grid-template-areas:   'b b b b b b b'
                           'b b b b b b b'
                           'o o o o o g g'
                           'o o o o o g g'
                           'c c . t u g g'
                           'y y . t u g g'
                           'y y r r r r r'
                           'k k k k k k k'
                           'k k k k k k k';
```

. means empty spaces



css Nav DropDwon

HTML

```
        <nav id="nav">
            <li>About me
                <ul>
                    <a>Go</a>
                    <a>Ea</a>
                    <a>ME</a>
                </ul>
            </li>
            <li>Cooking stuff</li>
            <li>traveling stuff</li>
            <li>Coding stuff</li>
            <li>Info</li>
        </nav>
```

css

```
nav > li > ul {
    display: none;
}
nav > li:hover > ul {  go to ul and do {} when I hover the li

    display: flex;
    flex-direction: column;
}
```

CSS text-shadow

```
text-shadow: h-shadow v-shadow blur-radius color;

text-shadow: 0vw 0 5vw red;
```

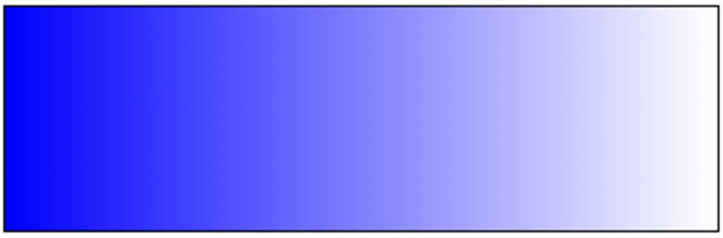
NAFAS

```
text-shadow: -1vw 0 2vw red,1vw 0 2vw yellow, 0 -1vw 2vw red, 0 1vw 2vw
yellow;
```

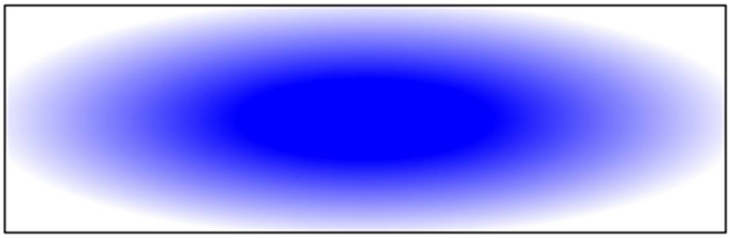
NAFAS

css gradient

```
background: linear-gradient(90deg, blue, white);
```



```
background: radial-gradient( blue 25%, white 75%);
```



CSS **transform** Property

The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew انحرف, etc., elements.

```
transform: none|transform-functions();
```

transform: rotate(20deg):

transform: skewY(20deg):

transform: scaleY(1.5):

https://www.w3schools.com/cssref/css3_pr_transform.asp

and we can do :
```
transition: transform 0.4s ease-out;
```

CSS Animations

CSS animations allows animation of most HTML elements without using JavaScript or Flash!

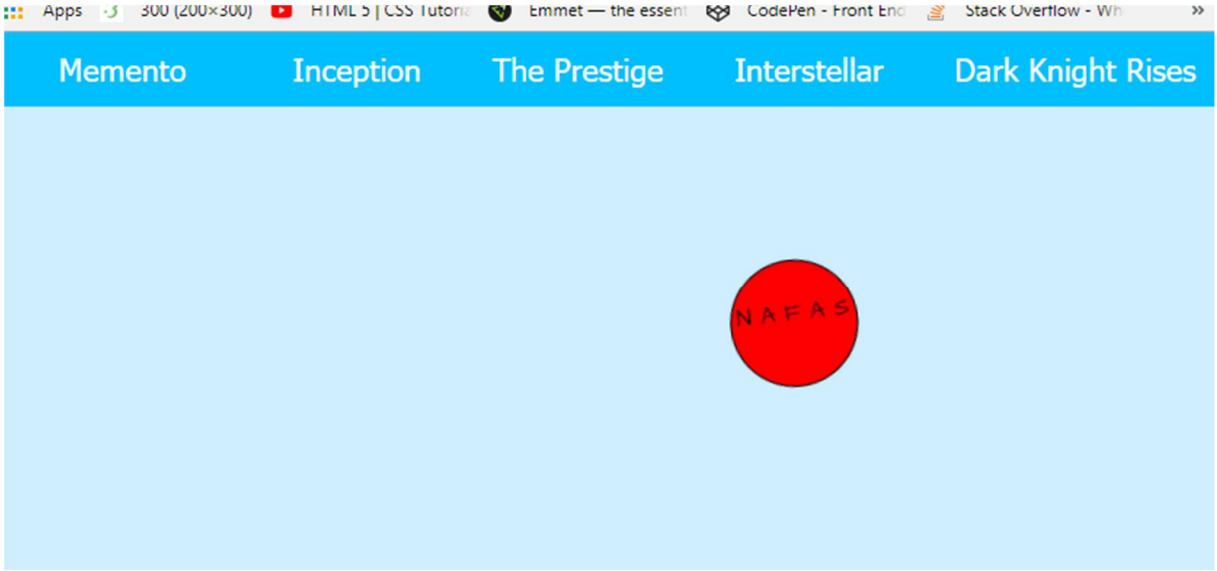
```
@keyframes VaribaleName {
  from {background: white;}
  to {background:black; width: 30vw;}
}

section div {
  border: 1px solid black;
  height: 10vw; width: 10vw;
  background: red;
  position: relative;
  top: 40%;


  animation-name: VaribaleName;
  animation-duration: 5s;
  animation-timing-function: linear;


}
```

```
@keyframes VaribaleName {
  0% {transform: translate(0vw,0) ; }
  25% {transform: translate(5vw,10vw) rotate(25deg); }
```

```
  50% {transform: translate(10vw,-10vw); }
  75% {transform: translate(30vw, 0) rotate(75deg); }
  100% {transform: translate(40vw, 5vw) rotate(-5deg); }
}
section:hover div {
  border: 1px solid black;
  height: 10vw; width: 10vw;
  background: red;
  position: relative;
  top: 40%;

  animation-name: VaribaleName;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 0.5s;
  animation-iteration-count: infinite;
  animation-direction: alternate;

  /*or animation: VaribaleName 5s  linear 0.5s infinite  alternate; *
}
```

https://www.w3schools.com/css/css3_animations.asp

```
@keyframes VaribaleName {
  0% {transform: translate(0vw,10vw) rotate(20deg) ; }
  25% {transform: translate(45vw,0vw); }
  50% {transform: translate(90vw,10vw) rotate(-20deg); }
  75% {transform: translate(45vw, 20vw); }
  100% {transform: translate(0vw, 10vw);}
}

section div {
  border: 1px solid black;
  height: 10vw; width: 10vw;
  border-radius: 50px;
  background: red;
  text-align: center;
  position: relative;
  top: 10%;


  animation-name: VaribaleName;
  animation-duration: 10s;
  animation-timing-function: linear;
  animation-delay: 0.5s;
  animation-iteration-count: infinite;
  /* animation-direction: alternate; */

  /*or animation: VaribaleName 5s  linear 0.5s infinite  alternate; */

}
```



HTML Basic Icons , Font Awesome

To use the Font Awesome icons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

You place Font Awesome icons by using the prefix fa and the icon's name.

The following code:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-
awesome.min.css">
</head>
<body>

<i class="fa fa-car"></i>
<i class="fa fa-car" style="font-size:48px;"></i>
<i class="fa fa-car" style="font-size:60px;color:red;"></i>

</body>
</html>
```

Results in:

HTML CSS root doesn’t include the

Important Note: the root doesn’t include the Body.

So do that:

```
:root {
    background: #cccccc;
    margin: 0;
    border: 1px solid black;
    padding: 0;
}
body {
    margin: 0;
    padding: 0;
}
```

CSS Flexbox here all explain

Before the Flexbox Layout module, there were four layout modes:

- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

To start using the Flexbox model:

1- you need to first define a flex container (Parent Element).

HTML:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

CSS:
The flex container becomes flexible by setting the `display` property to

```
.flex-container {
  display: flex;
 }
```

2-flex items

The **direct** child elements of a flex container automatically becomes flexible (flex) items.

HTML:

```
<div class="flex-container">      flex container
  <div>1</div>        flex item
  <div>2</div>        flex item
  <div>3</div>        flex item
</div>
```

So the whole code is :

```
<body>

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>

</body>
```

```
.flex-container {
  display: flex;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
```

.....

The flex container properties are:

- flex-direction
- flex-wrap لف
- flex-flow
- justify-content
- align محاذاة اصطف -items
- align-content

The flex-direction Property

The `flex-direction` property defines in which direction the container wants to stack the flex items.

```
.flex-container {                    …… flex-direction: column;
  display: flex;
  flex-direction: column;
  background-color: DodgerBlue;
}
```


```
.flex-container {     ….. flex-direction: column-reverse;
  display: flex;
  flex-direction: column-reverse;
  background-color: DodgerBlue;
}
```


```
.flex-container {     ….. flex-direction: row; is as Default
    display: flex;
```

```
    flex-direction: row;
    background-color: DodgerBlue;
  }
```


```
.flex-container {    …… flex-direction: row-reverse;
  display: flex;
  flex-direction: row-reverse;
  background-color: DodgerBlue;
}
```

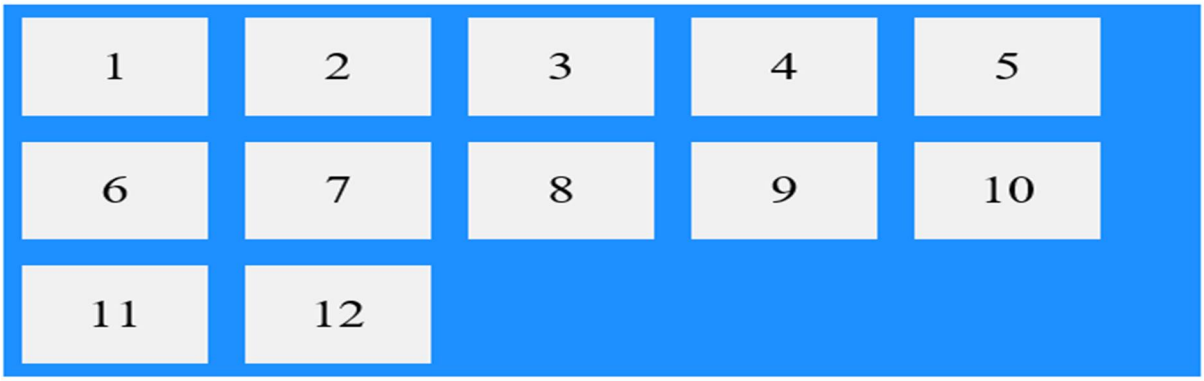
The "flex-direction: row-reverse;" stacks the flex items horizontally (but from right to left):

The flex-wrap Property

The `flex-wrap` لف property specifies whether the flex items should wrap or not.

```
.flex-container {     …. flex-wrap: wrap;
  display: flex;
  flex-wrap: wrap;
  background-color: DodgerBlue;
}
```

The "flex-wrap: wrap;" specifies that the flex items will wrap if necessary:



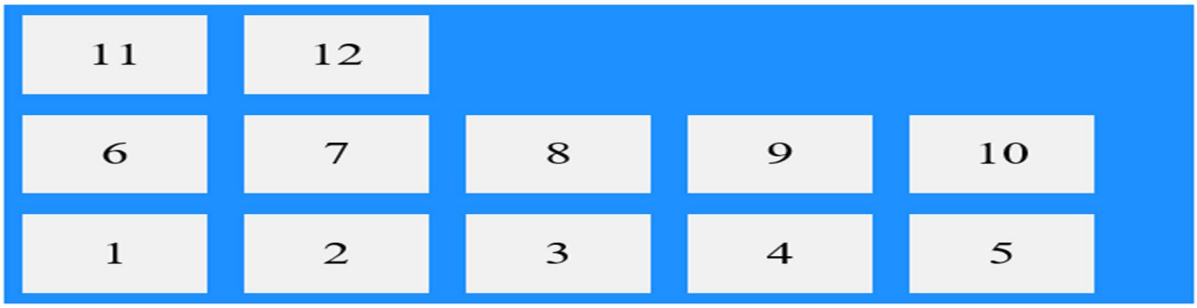

```
.flex-container {           ….. flex-wrap: nowrap;
  display: flex;
```

```
  flex-wrap: nowrap;
  background-color: DodgerBlue;
}
```

The "flex-wrap: nowrap;" specifies that the flex items will not wrap (this is default):


```
.flex-container {                    ….. flex-wrap: wrap-reverse;
  display: flex;
  flex-wrap: wrap-reverse;
  background-color: DodgerBlue;
}
```

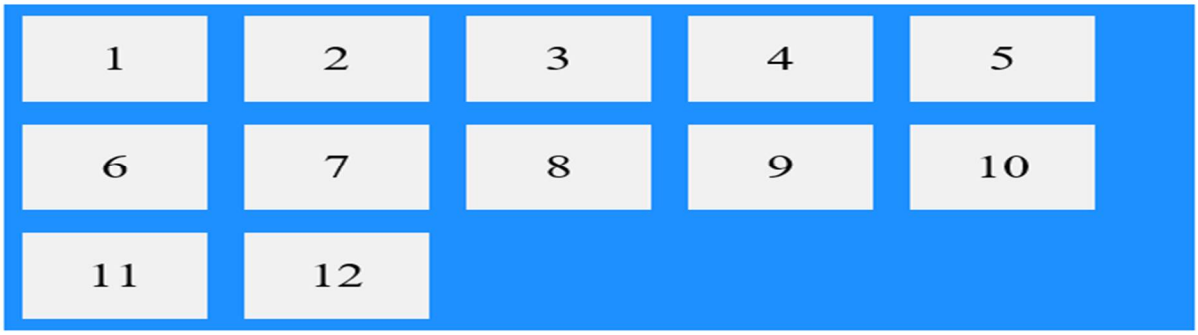
The "flex-wrap: wrap-reverse;" specifies that the flex items will wrap if necessary, in reverse order:



The flex-flow Property

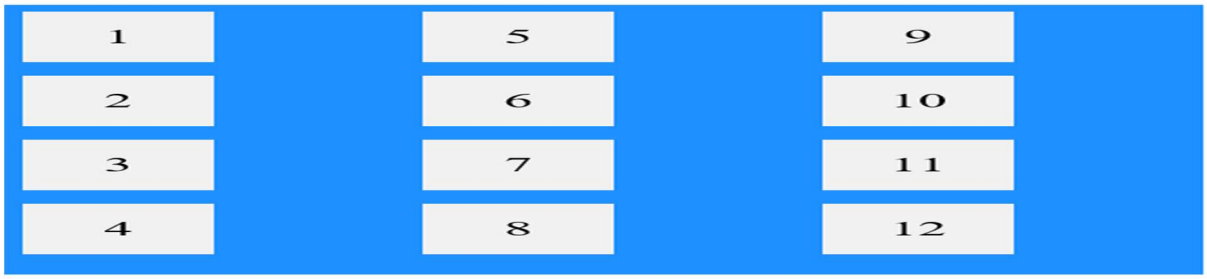
The `flex-flow` property is a shorthand property for setting both the `flex-direction` and `flex-wrap` properties.

```
.flex-container {                … flex-flow: row wrap;
  display: flex;
  flex-flow: row wrap;
  background-color: DodgerBlue;
}
```




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

```
  flex-flow: column wrap;
  background-color: DodgerBlue;
  height: 400px;                    ….. Important, try it without it
}
```



The justify-content Property

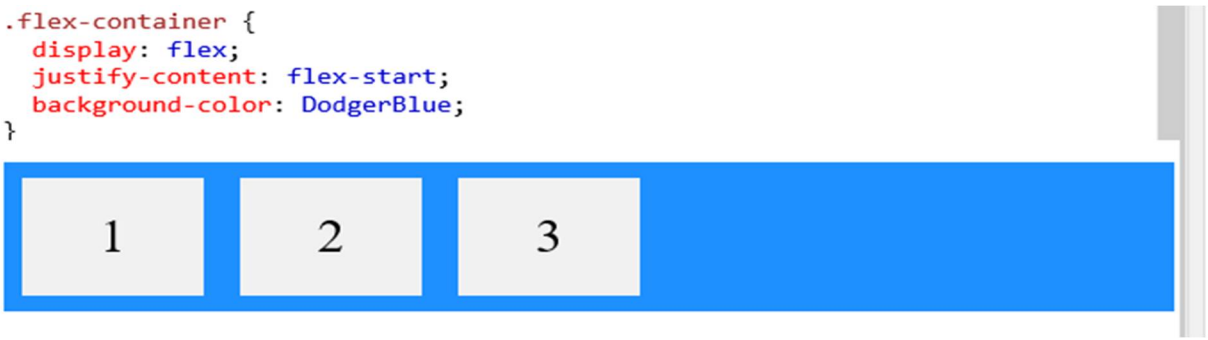
The `justify-content` property is used to align the flex items on X-axis.

The *center* value aligns the flex items at the center of the container:

```
.flex-container {
  display: flex;
  justify-content: center;
  background-color: DodgerBlue;
}
```

The *flex-start* value aligns the flex items at the beginning of the container (this is default):

```
.flex-container {
  display: flex;
  justify-content: flex-start;
  background-color: DodgerBlue;
}
```



The *flex-end* value aligns the flex items at the end of the container:

```
.flex-container {
  display: flex;
  justify-content: flex-end;
  background-color: DodgerBlue;
}
```

The *space-around* value displays the flex items with space before, between, and after the lines:

```
.flex-container {
  display: flex;
  justify-content: space-around;
  background-color: DodgerBlue;
}
```

The *space-between* value displays the flex items with space between the lines:

```
.flex-container {
  display: flex;
  justify-content: space-between;
  background-color: DodgerBlue;
}
```

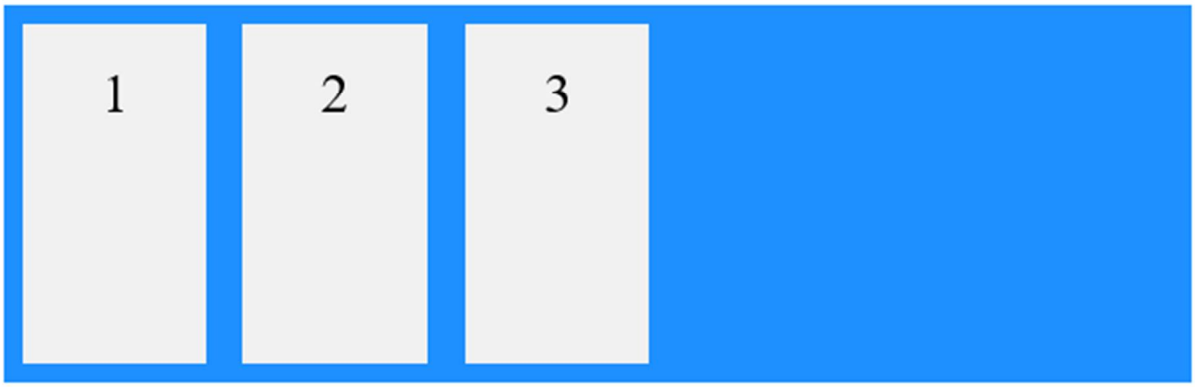
The align-items Property

The `align-items` property is used to align the flex items vertically on Y-axis.
عموديا

The *stretch* value stretches the flex items to fill the container (this is default):

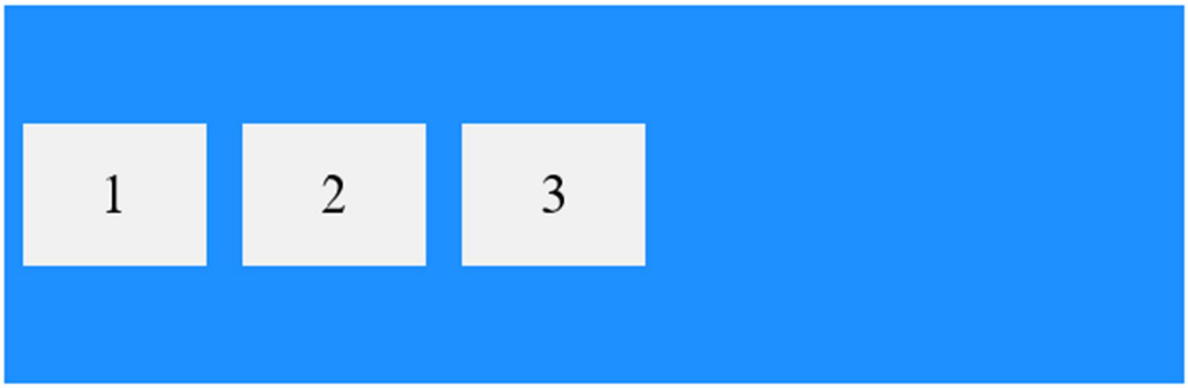
stretch تمتد

```
.flex-container {
  display: flex;
  height: 200px;
  align-items: stretch;
  background-color: DodgerBlue;
}
```



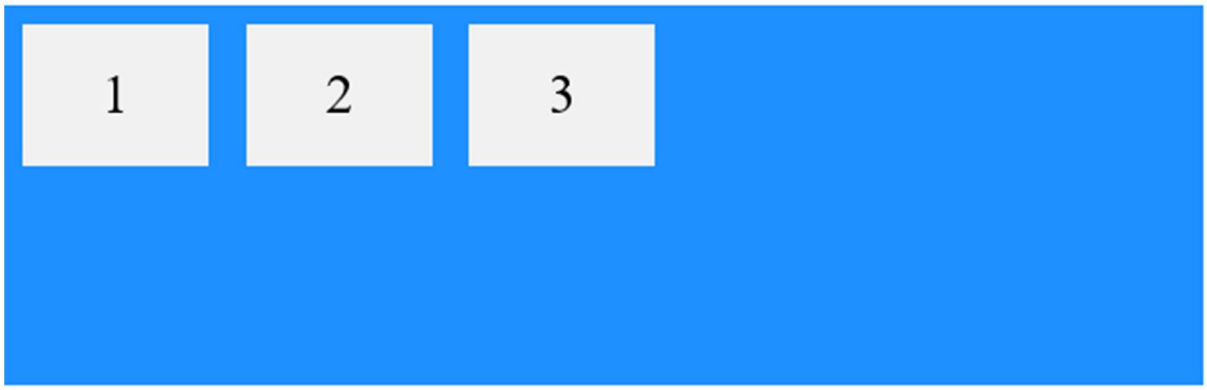
The *center* value aligns the flex items in the middle of the container:

```
.flex-container {
  display: flex;
  height: 200px;
  align-items: center;
  background-color: DodgerBlue;
}
```



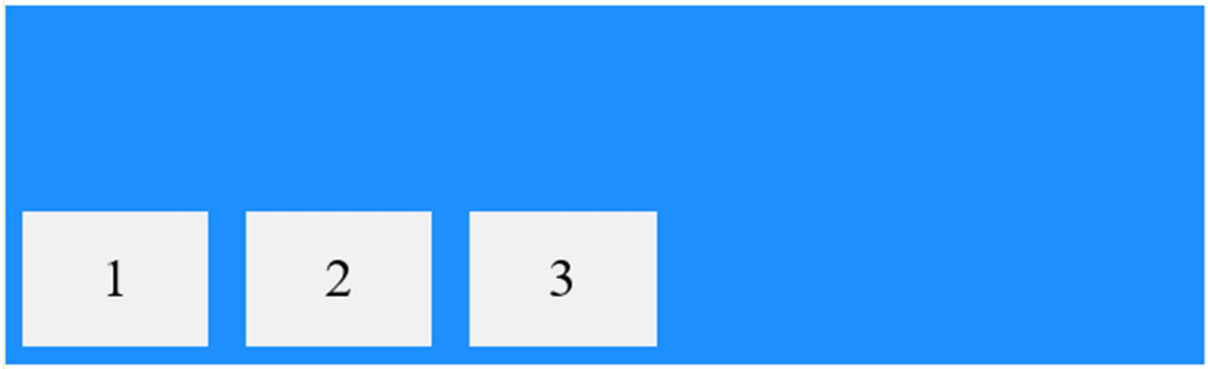
The *flex-start* value aligns the flex items at the top of the container:

```
.flex-container {
  display: flex;
  height: 200px;
  align-items: flex-start;
  background-color: DodgerBlue;
}
```



The *flex-end* value aligns the flex items at the bottom of the container:

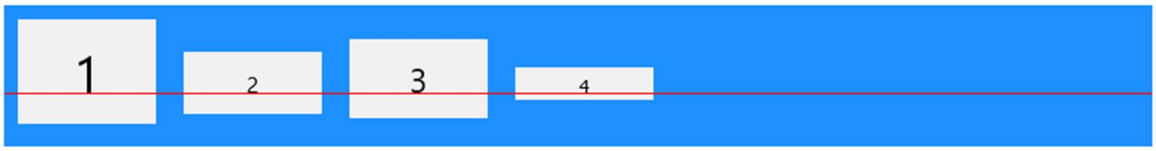
```
.flex-container {
  display: flex;
  height: 200px;
  align-items: flex-end;
  background-color: DodgerBlue;
}
```



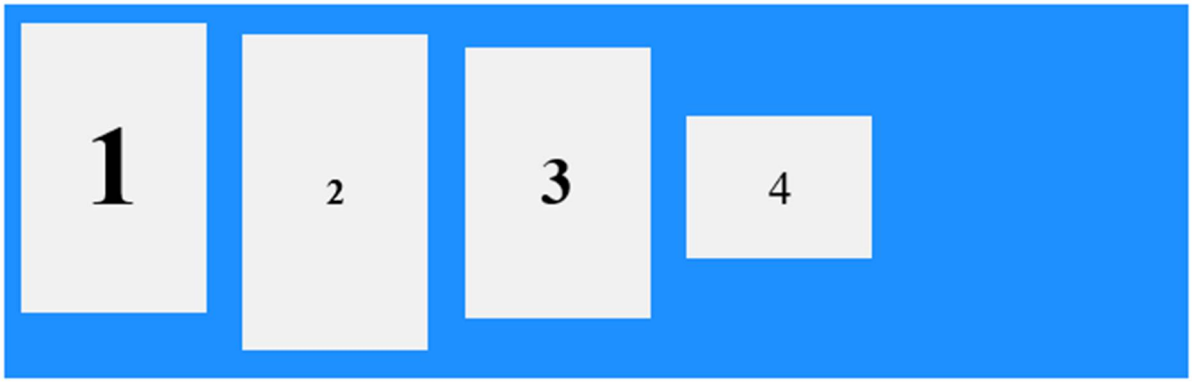
The *baseline* value aligns the flex items such as their baselines aligns:

```
.flex-container {
  display: flex;
  height: 200px;
  align-items: baseline;
}
```

Note: the example uses different font-size to demonstrate that the items gets aligned by the text baseline:



```
.flex-container {
  display: flex;
  height: 200px;
  align-items: baseline;
  background-color: DodgerBlue;
}
```



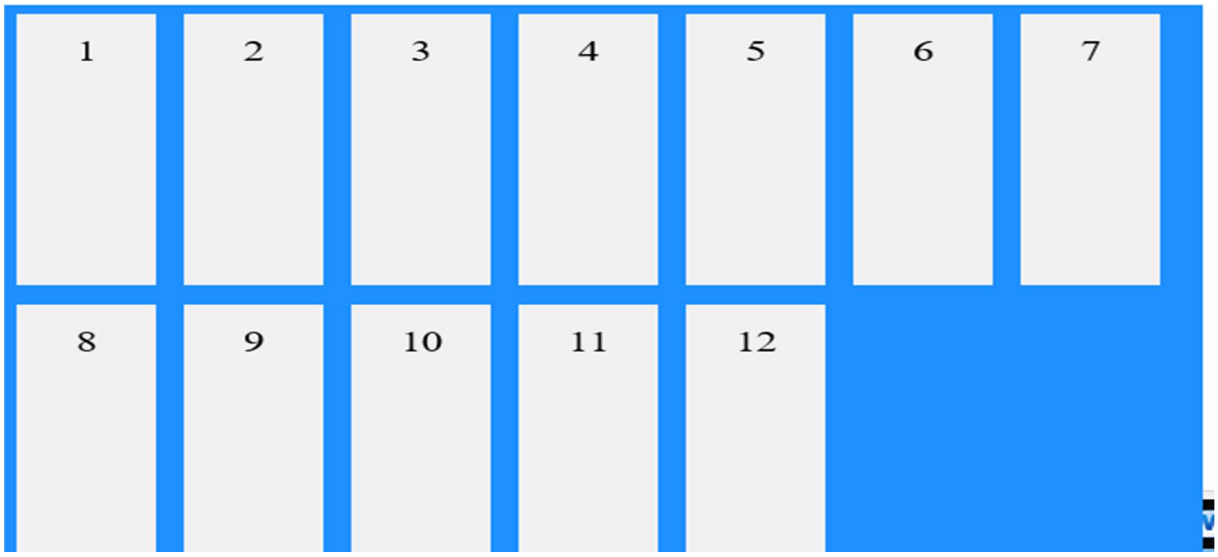
The align-content Property

The `align-content` property is used to align the flex lines (not the items, but the line of items).

In these examples we use a 600 pixels high container, with the flex-wrap property set to *wrap*, to better demonstrate the `align-content` property.

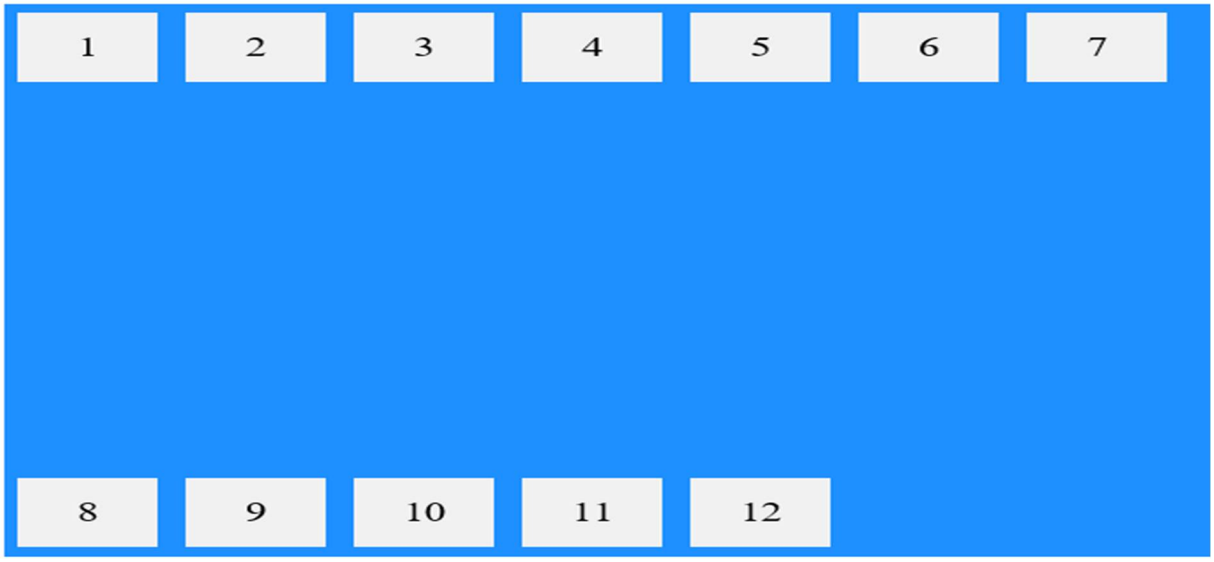
The *stretch* value stretches the flex lines to take up the remaining space (this is default):

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: stretch;
  background-color: DodgerBlue;
}
```



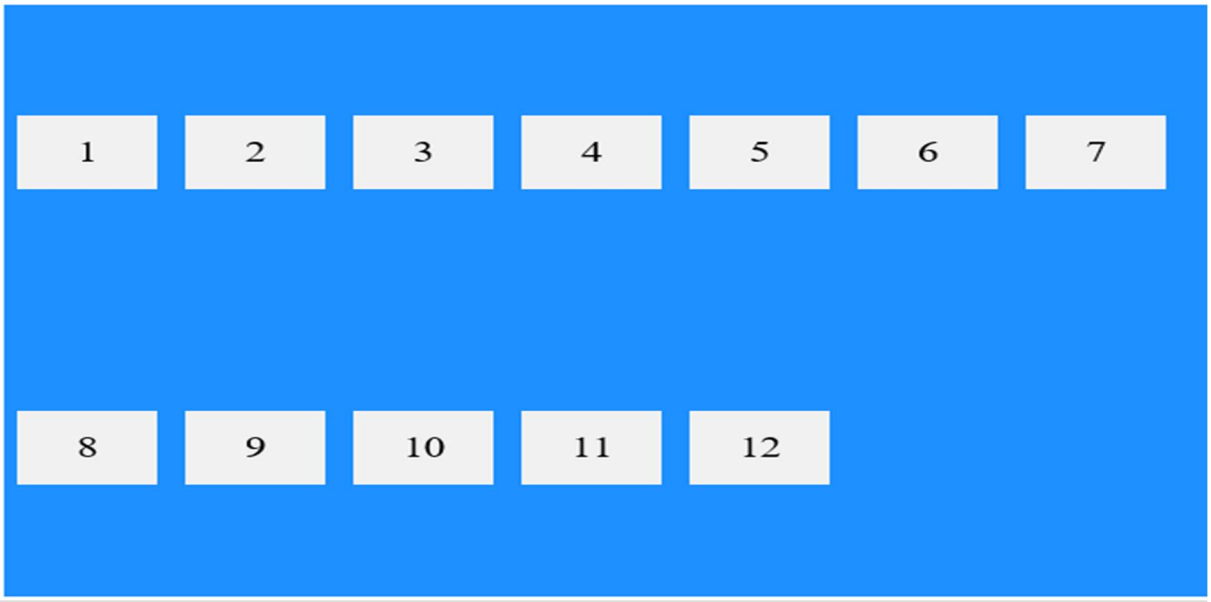
The *space-between* value displays the flex lines with equal space between them:

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: space-between;
  background-color: DodgerBlue;
}
```



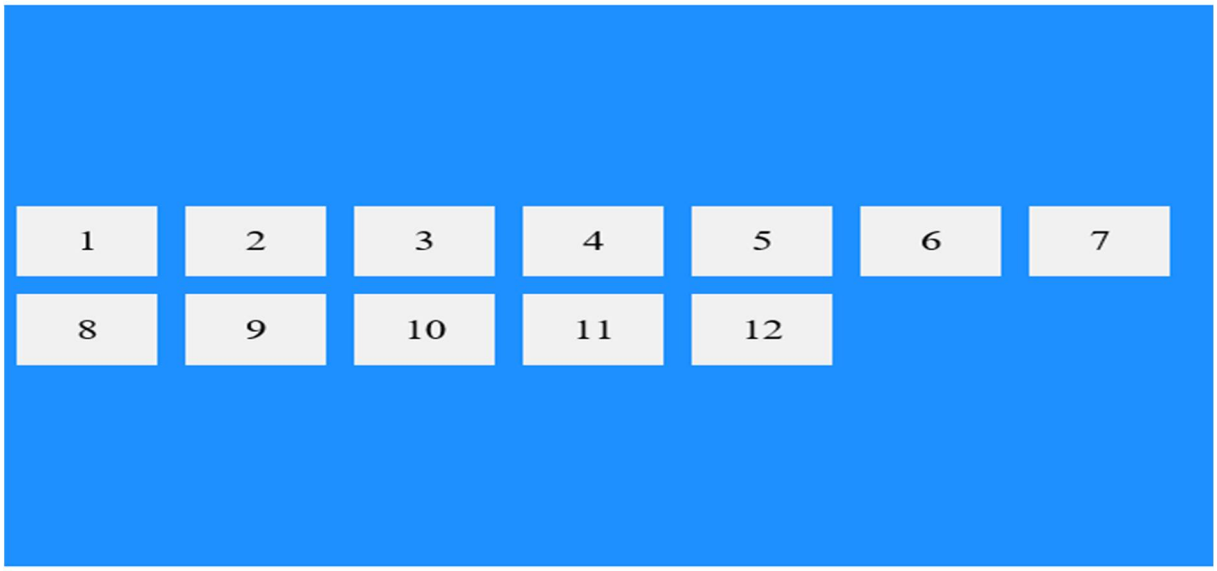
The *space-around* value displays the flex lines with space before, between, and after them:

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: space-around;
  background-color: DodgerBlue;
}
```



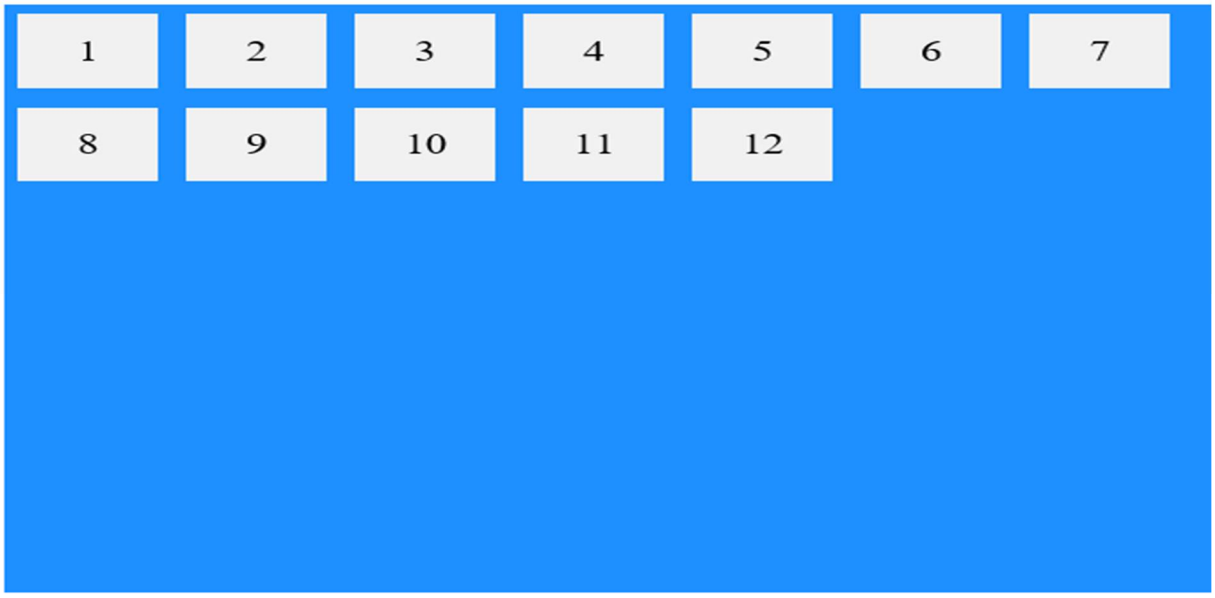
The *center* value displays display the flex lines in the middle of the container:

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: center;
}
```



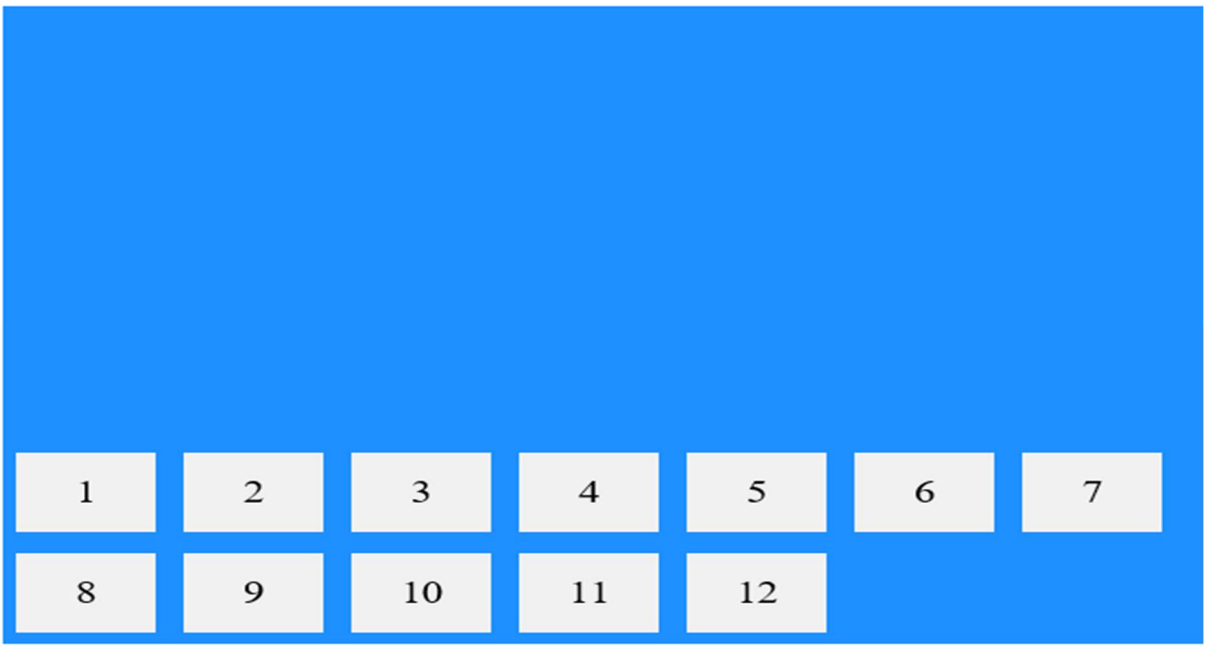
The *flex-start* value displays the flex lines at the start of the container:

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: flex-start;
}
```



The *flex-end* value displays the flex lines at the end of the container:

```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: flex-end;
}
```



Perfect Centering

In the following example we will solve a very common style problem: perfect centering.

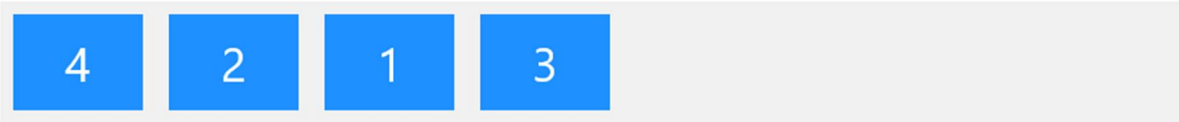
```
.flex-container {
  display: flex;
  height: 300px;
  justify-content:center;     ….x-axis
  align-items: center;        ….y-axis
}
```

……..

The flex items properties are:

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

The order property specifies the order of the flex items.



The first flex item in the code does not have to appear as the first item in the layout.

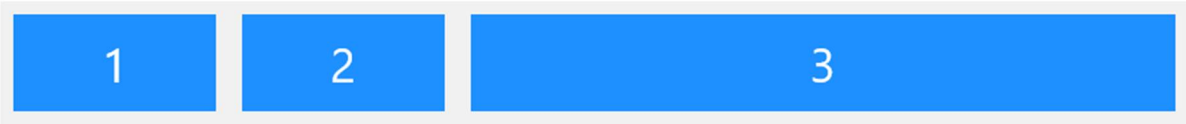
The order value must be a number, default value is 0.

Example

The *order* property can change the order of the flex items:

```
<div class="flex-container">
  <div style="order: 3">1</div>
  <div style="order: 2">2</div>
  <div style="order: 4">3</div>
  <div style="order: 1">4</div>
</div>
```

The flex-grow property specifies how much a flex item will grow relative to the rest of the flex items. grow تنمو



The value must be a number, default value is 0.

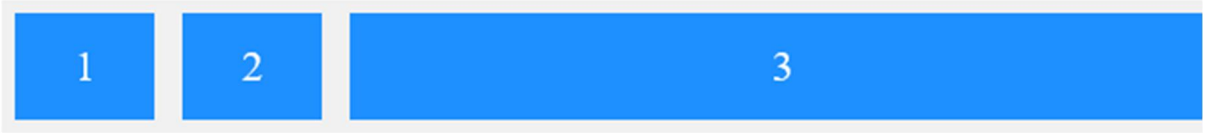
Example

Make the third flex item grow eight times faster than the other flex items:

```
<div class="flex-container">
  <div style="flex-grow: 1">1</div>
  <div style="flex-grow: 1">2</div>
  <div style="flex-grow: 8">3</div>
</div>
```

Or

```
<div class="flex-container">
  <div >1</div>
  <div >2</div>
  <div style="flex-grow: 1">3</div>  … here this div will take all the rest
</div>
```




```
#main div {
  flex-grow: 1;     .... = grow 1 , all items will share the rest space
}
```

The flex-shrink Property

The `flex-shrink` property specifies how much a flex item will shrink relative to the rest of the flex items. shrink تقلص

خاصية الانكماش تحدد مقدار تقلص عنصر المرن بالنسبة لبقية العناصر المرنة.

The value must be a number, default value is 1.

0 no shrink , it will be its normal size

1 shrink, it will shrink of 1 unite and it will be smaller size

2 shrink, it will shrink of 2 unite and it will be more smaller size

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-shrink: 0">3</div>
  <div style="flex-shrink: 1">4</div>
  <div style="flex-shrink: 2">5</div>
  <div style="flex-shrink: 3">6</div>
  <div>7</div>
  <div>8</div>
  <div>9</div>
  <div>10</div>
</div>
```

The flex-basis Property

The `flex-basis` property specifies the initial length of a flex item.

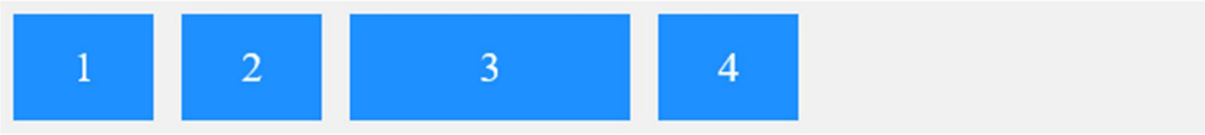
```
HTML:


<p>Set the initial length of the third flex item to 200 pixels:</p>

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div style="flex-basis:200px">3</div>
  <div>4</div>
</div>

Css:
.flex-container > div {
  background-color: DodgerBlue;
  color: white;
  width: 100px;
  margin: 10px;
  text-align: center;
  line-height: 75px;
  font-size: 30px;
}
```

Set the initial length of the third flex item to 200 pixels:

The flex Property

The `flex` property is a shorthand property for the `flex-grow`, `flex-shrink`, and `flex-basis` properties

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
   <div style="flex: 0 0 200px">3</div>  … grow=0 no grow,shrink=0 no shrink it
                                          will be its normal size , basis=200ox

  <div>4</div>
</div>
```



The align-self Property

The `align-self` property specifies the alignment المحاذاة لfor the selected item inside the flexible container.

The `align-self` property overridesلغى the default alignment set by the container's `align-items` property.

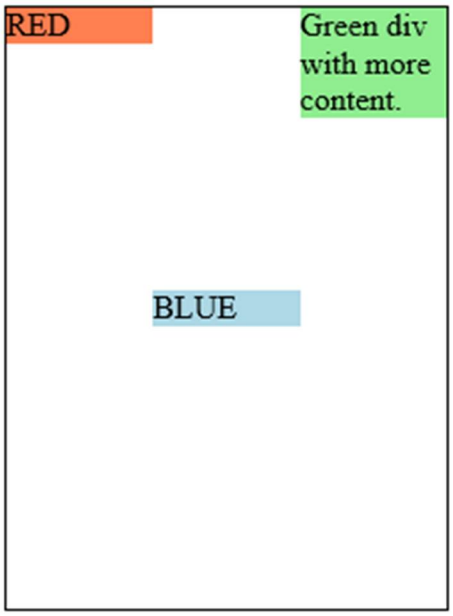
```
align-self: auto|stretch|center|flex-start|flex-end|baseline|initial|inherit;
```

Value	Description
auto	Default. The element inherits its parent container's align-items property, or "stretch" if it has no parent container
stretch	The element is positioned to fit the container
center	The element is positioned at the center of the container
flex-start	The element is positioned at the beginning of the container
flex-end	The element is positioned at the end of the container
baseline	The element is positioned at the baseline of the container
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element. Read about *inherit*

Ex.

```
<div id="main">
  <div style="background-color:coral;">RED</div>
  <div style="background-color:lightblue;" id="myBlueDiv">BLUE</div>
  <div style="background-color:lightgreen;">Green div with more content.</div>
</div>
#main {
  width: 220px;
  height: 300px;
  border: 1px solid black;
  display: flex;
  align-items: flex-start;
}
```

```
#main div {
  flex: 1;    .... = grow 1 , all items will share the rest space
}

#myBlueDiv {
  align-self: center;
}
```



.....

The Summary Flex-container + Flex-items

```
Flex-container
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    align-items: flex-start;
    flex-wrap: wrap;
```

```
Flex-items
    flex-basis: 20%;
    flex-grow: 0.5;
    flex-shrink: 1;
    align-self: center;
    order: 2;
```

shortcut:

```
in Flex-container
   flex-direction: row;
    flex-wrap: wrap
```

shortcut >>>> flex-flow: row warp;

```
in Flex-items
    flex-basis: 20%;
    flex-grow: 0.5;
    flex-shrink: 1;
```

shortcut >>>> flex: 0.5 1 20%;

css Example

Css Example

css Example

Css Example

css Example

Css Example

css Example

Css Example

css Example

Css Example

css Example

Css Example

css box-shadow

Html Block, inline, Inline-Block element ????

<img> <button> are Innline-Block element ?????

CSS LINK

<link rel="stylesheet" type="text/css" href="style.css">