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(){});

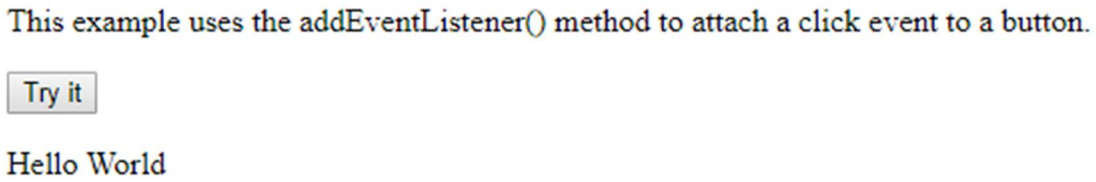
- **Tip:** Use the removeEventListener() method to remove an event handler that has been attached with the addEventListener() method.

-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



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

```css
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

```css
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

```html
    <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

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

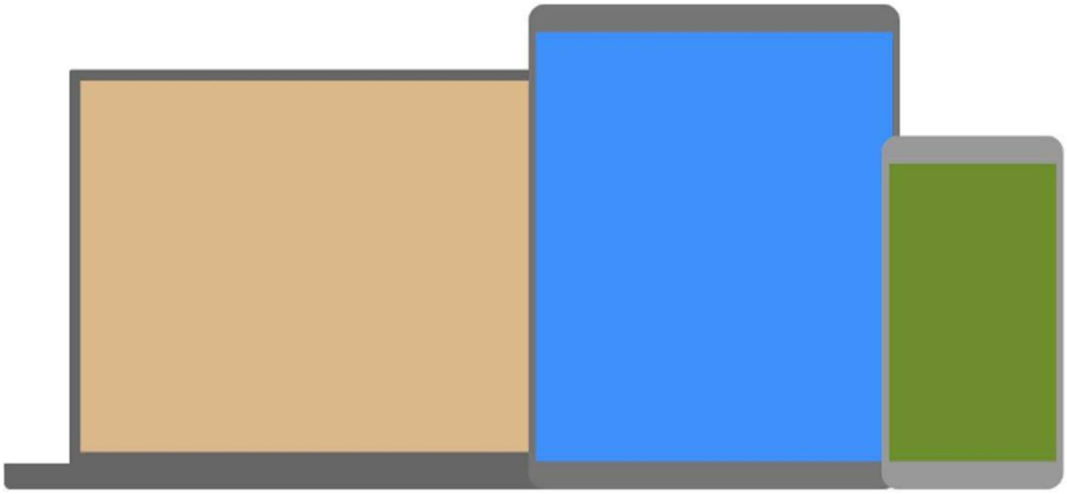
that will effect on inline elements and inside of Block elements, not on Block elements.

So the block elements will be not in center

Css Media queries

It uses the `@media` rule to include a block of CSS properties only if a certain condition is true.

Media queries are a popular technique for delivering a tailored style sheet to different devices. To demonstrate a simple example, we can change the background color for different devices:



```
/* On screens that are 992px or less, set the background color to blue */
@media screen and (max-width: 992px) {
  body {
    background-color: blue;
  }
}
```

```
/* On screens that are 600px or less, set the background color to olive */
@media screen and (max-width: 600px) {
  body {
    background-color: olive;
  }
}
```

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…

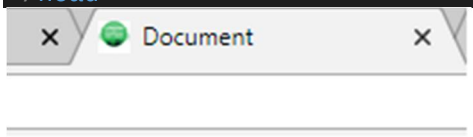
it works with div… so the solution is to set the image as background for div =➔ background-image: url(./images/goettingen.jpg);

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

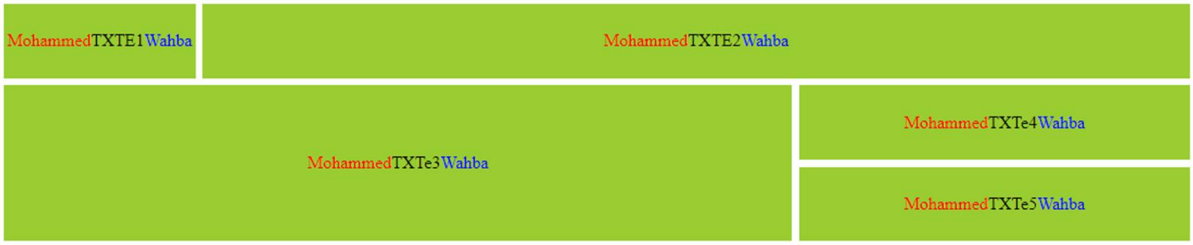
The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

```
.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

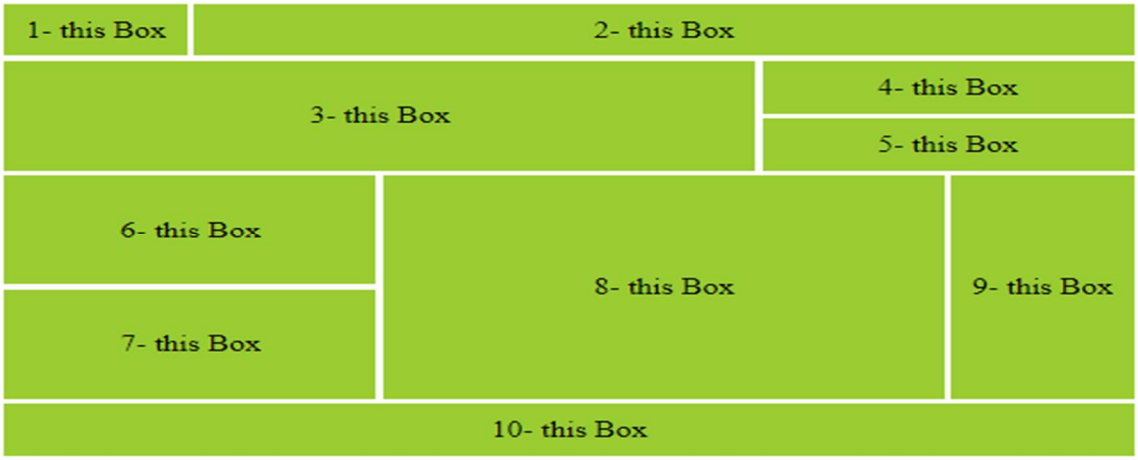
it can not to copy and mark it

```
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

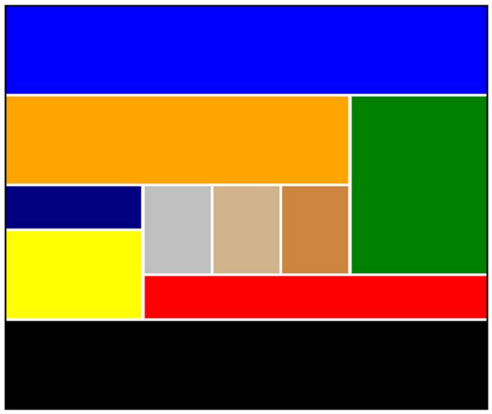
Css Variables, we define them either in :root or body.

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.goooodkdmdn,
}
.it8 {
    background-color: var(--varible-myBlue)
}
```

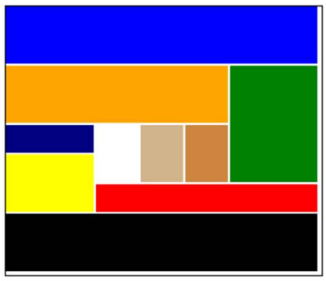
HTML CSS Grafic Grid

```css
.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;
}
```

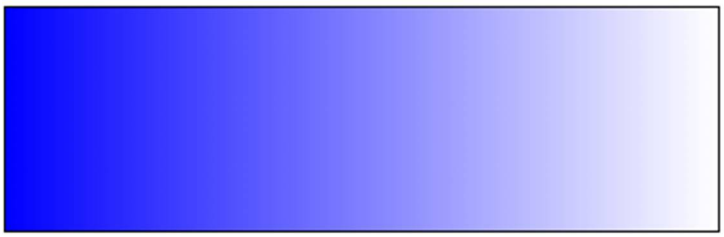
CSS text-shadow

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

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

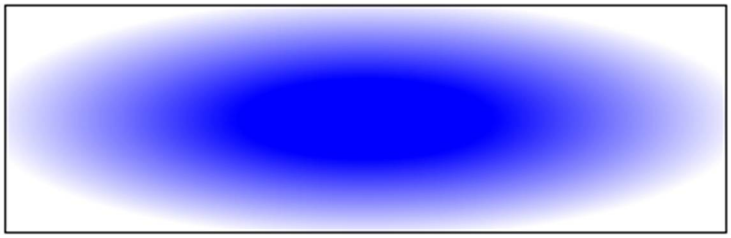

```
text-shadow: -1vw 0 2vw red,1vw 0 2vw yellow, 0 -1vw 2vw red, 0 1vw 2vw yellow;
```

css gradient

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



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



css **transform** Property

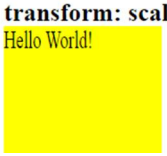
The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew انحرف, etc., elements.

```
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!

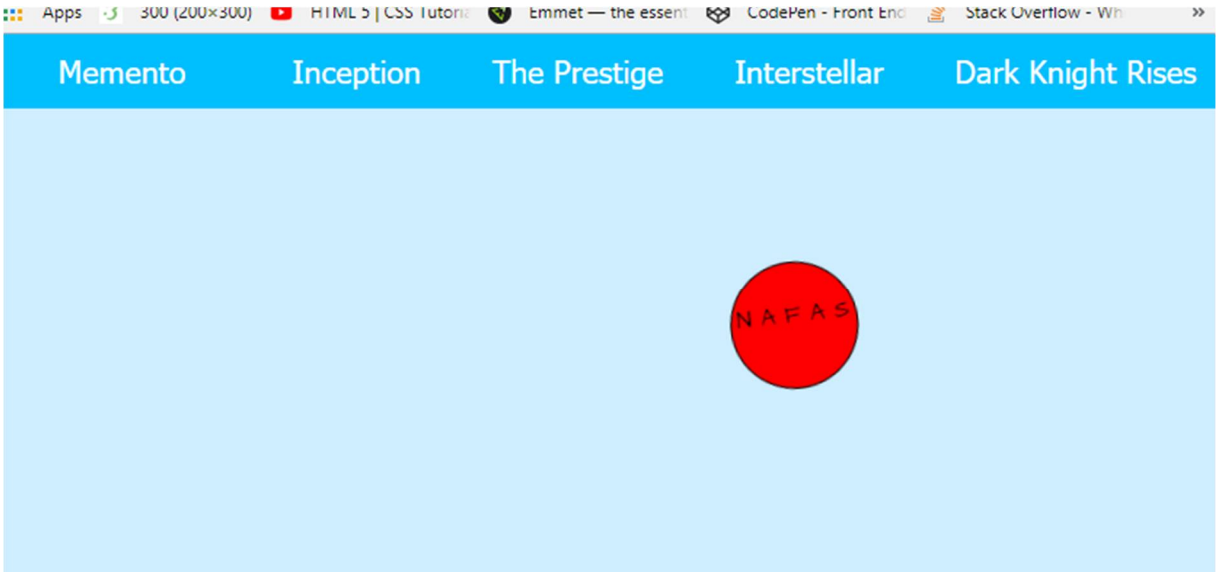
```
@keyframes VaribaleName {
  from {background: white;}
  to {background:black; width: 30vw;}
}

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;
}
```

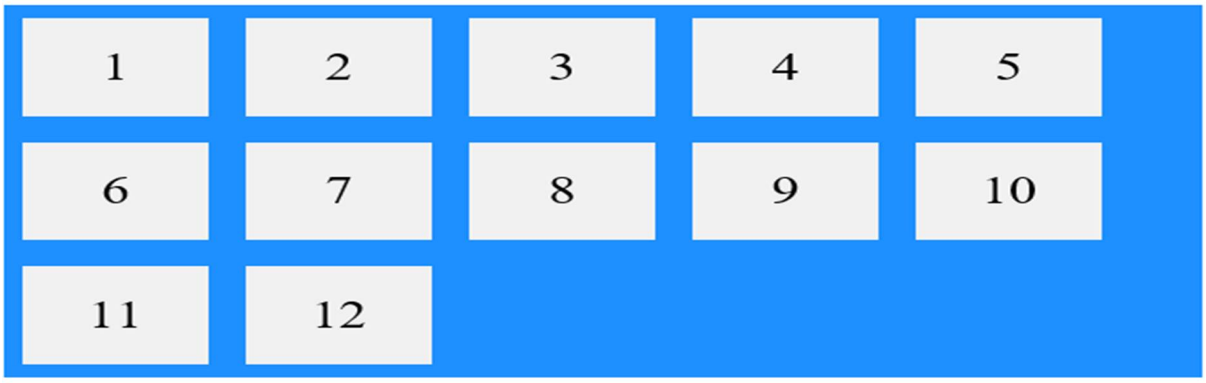
The "flex-direction: row-reverse;" stacks the flex items horizontally (but from right to left):

The flex-wrap Property

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

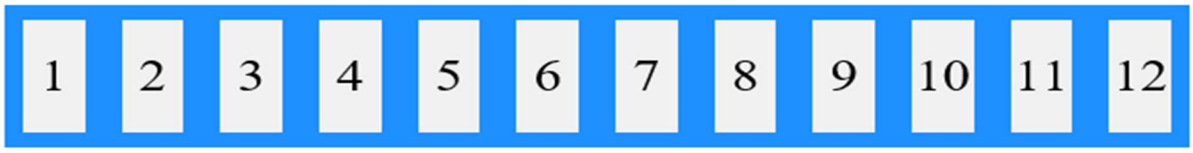
```
.flex-container {     …. flex-wrap: wrap;
  display: flex;
  flex-wrap: wrap;
  background-color: DodgerBlue;
}
```

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



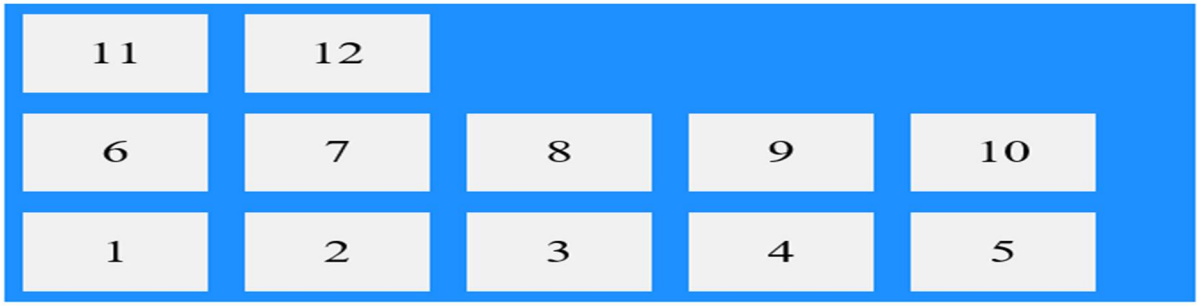

```
.flex-container {           ….. flex-wrap: nowrap;
  display: flex;
  flex-wrap: nowrap;
  background-color: DodgerBlue;
}
```

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;
}
```

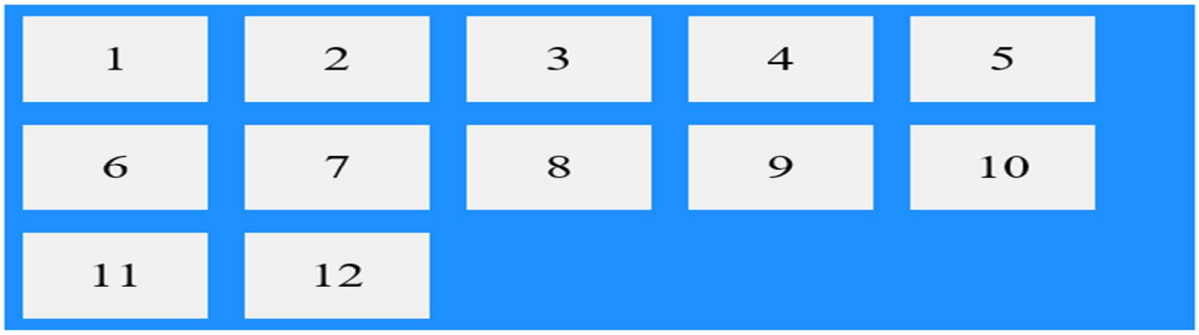
The "flex-wrap: wrap-reverse;" specifies that the flex items will wrap if necessary, in reverse order:



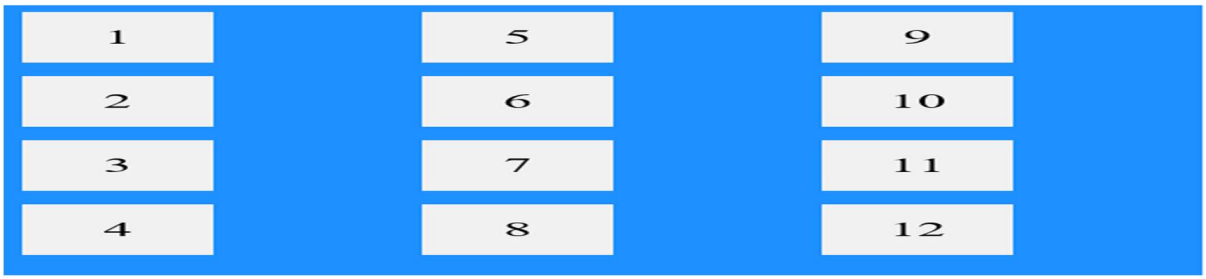
The flex-flow Property

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):

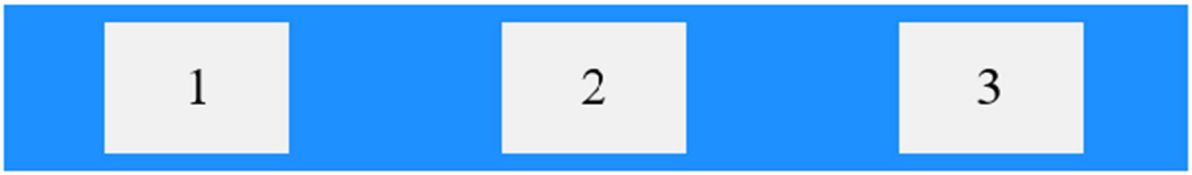
```
.flex-container {
  display: flex;
  justify-content: flex-start;
  background-color: DodgerBlue;
}
```

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;
}
```

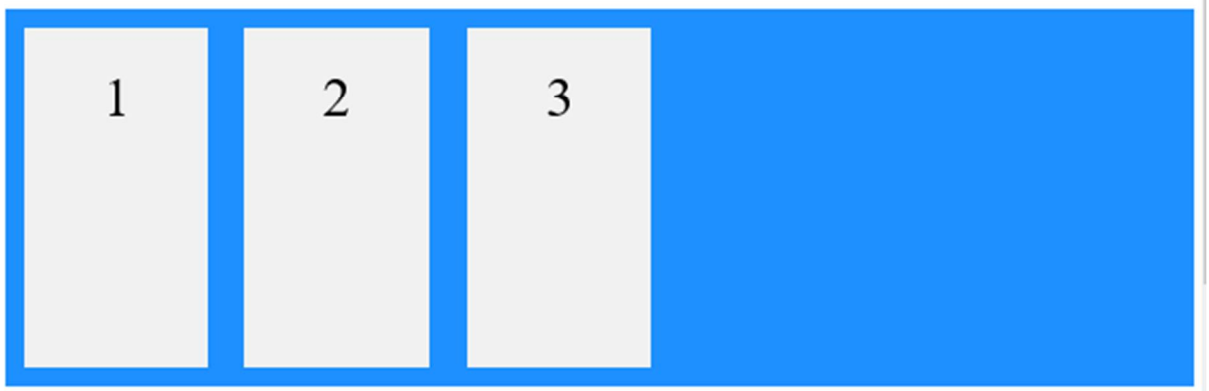
The align-items Property

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):

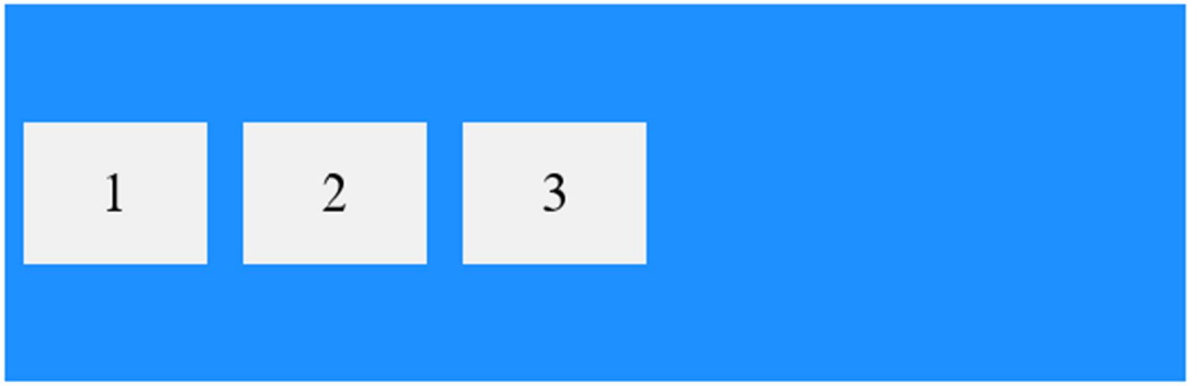
stretch تمتد

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



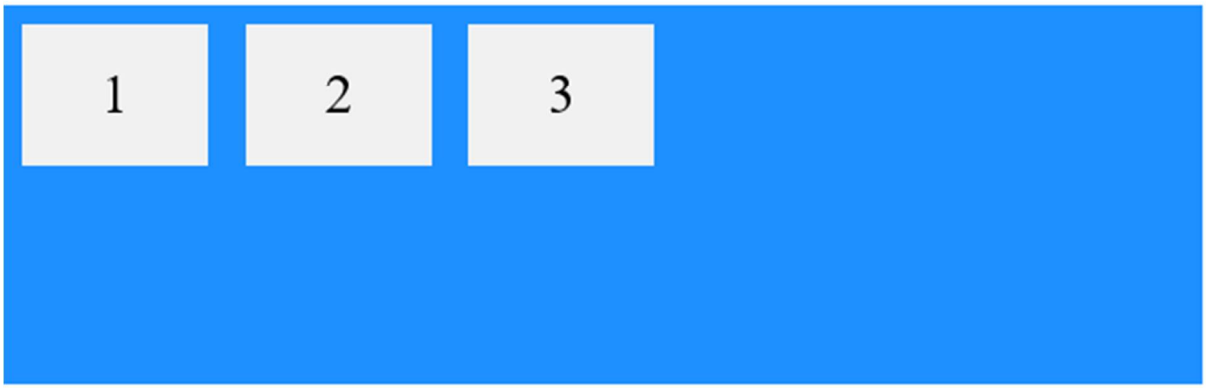
The *center* value aligns the flex items in the middle of the container:

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



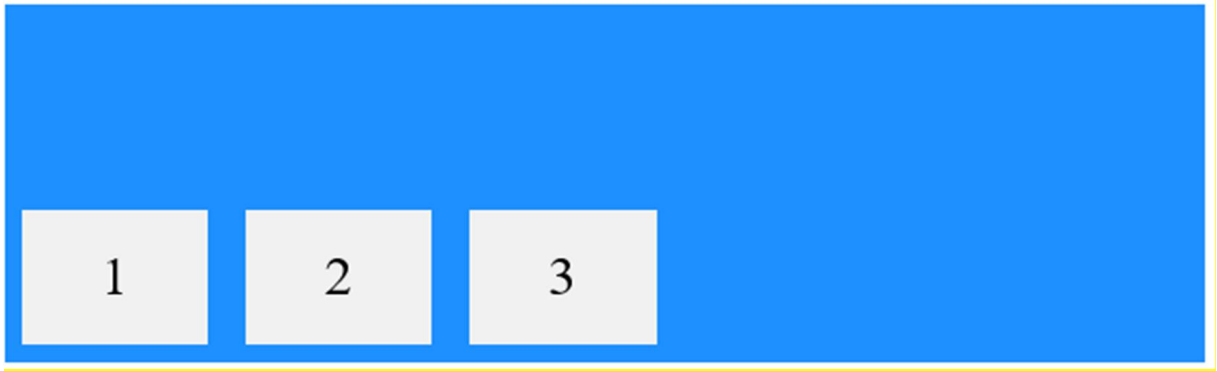
The *flex-start* value aligns the flex items at the top of the container:

```css
.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:

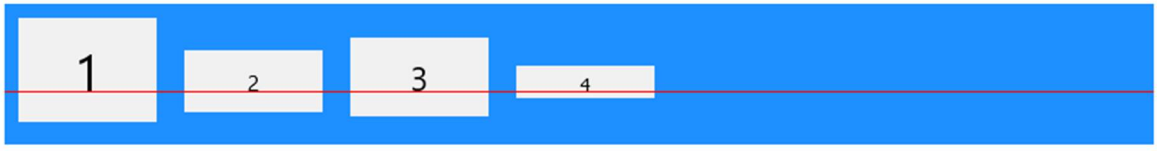
```css
.flex-container {
  display: flex;
  height: 200px;
  align-items: flex-end;
  background-color: DodgerBlue;
}
```



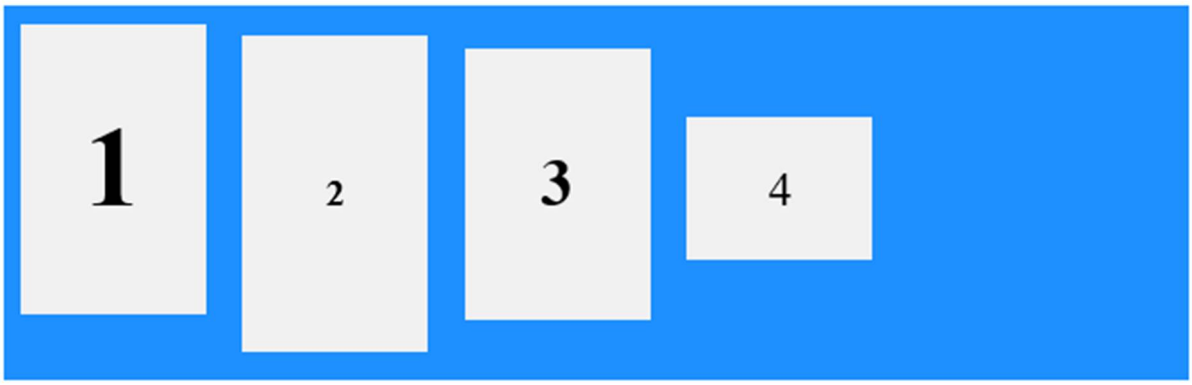
The *baseline* value aligns the flex items such as their baselines aligns:

```
.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;
}
```



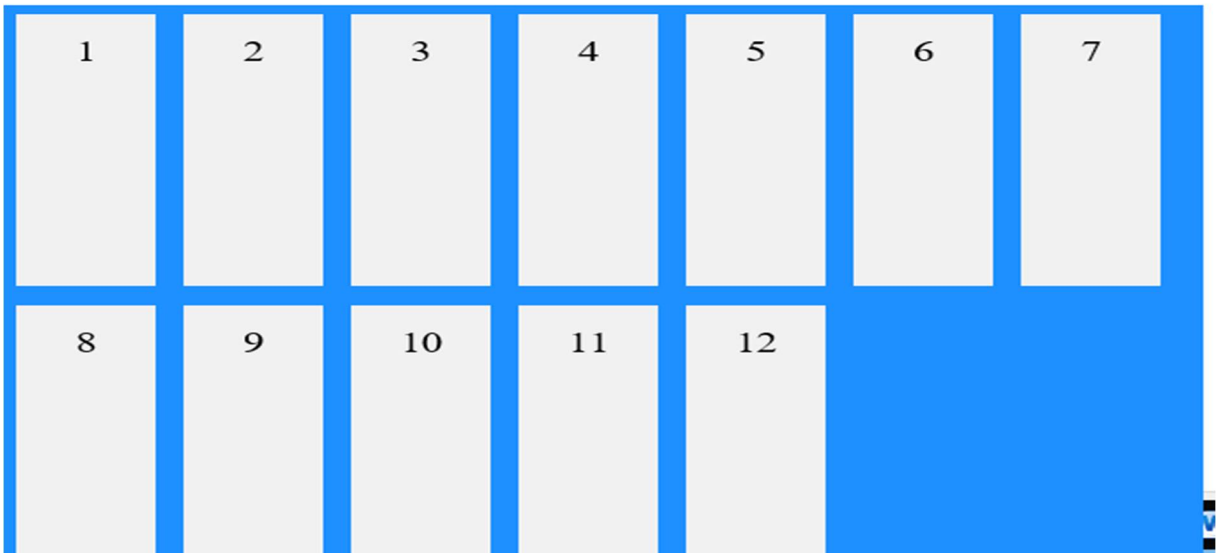
The align-content Property

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.

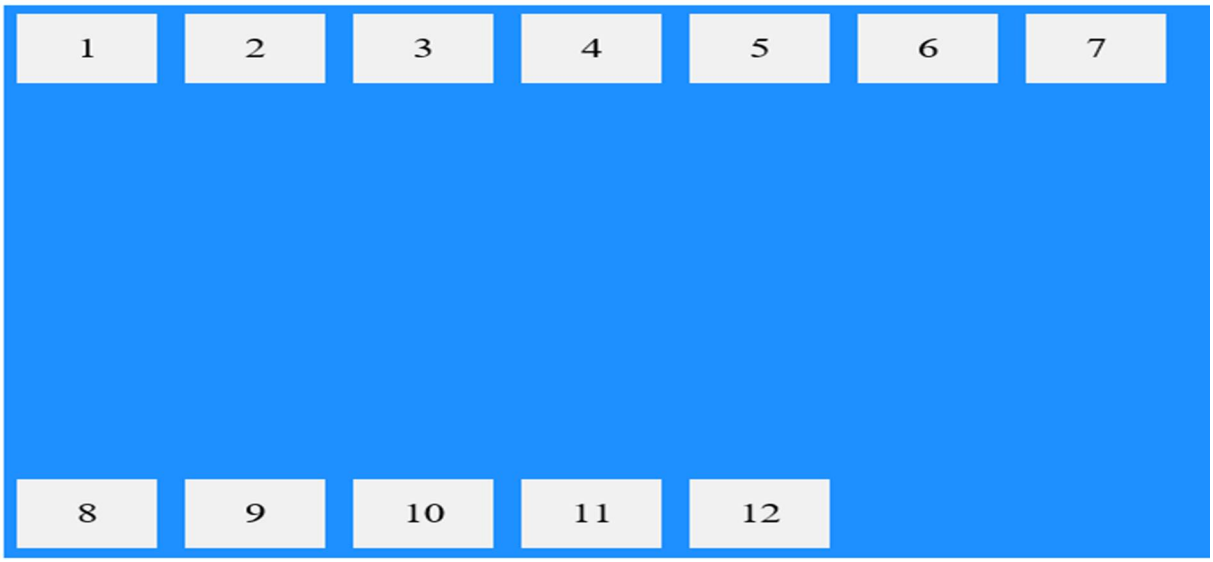
The *stretch* value stretches the flex lines to take up the remaining space (this is default):

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



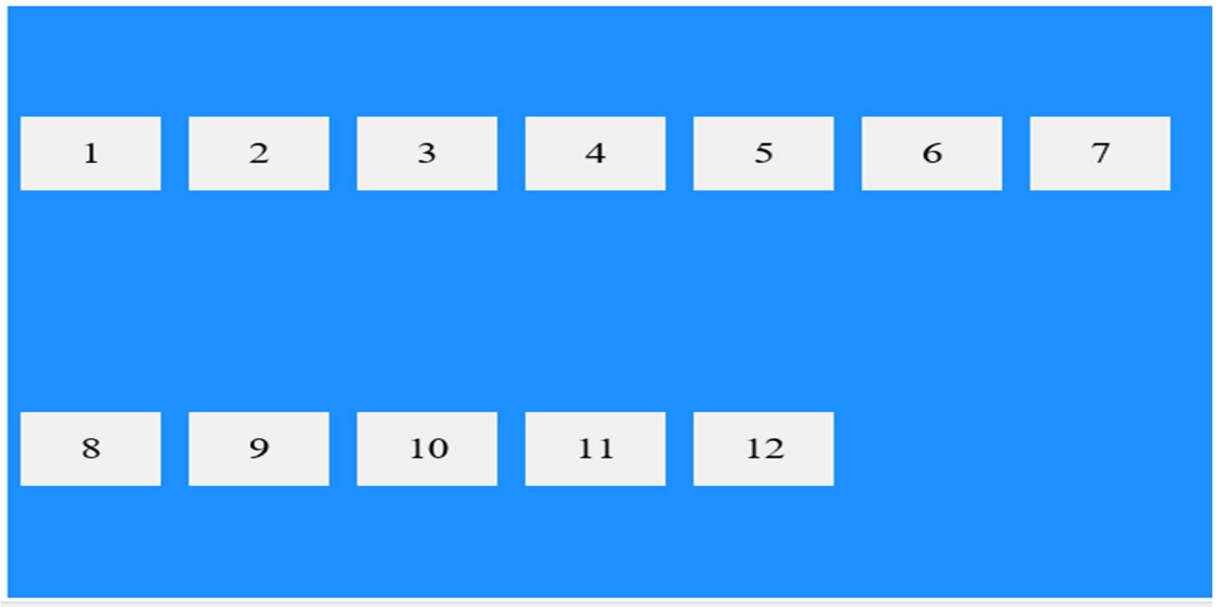
The *space-between* value displays the flex lines with equal space between them:

```
.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:

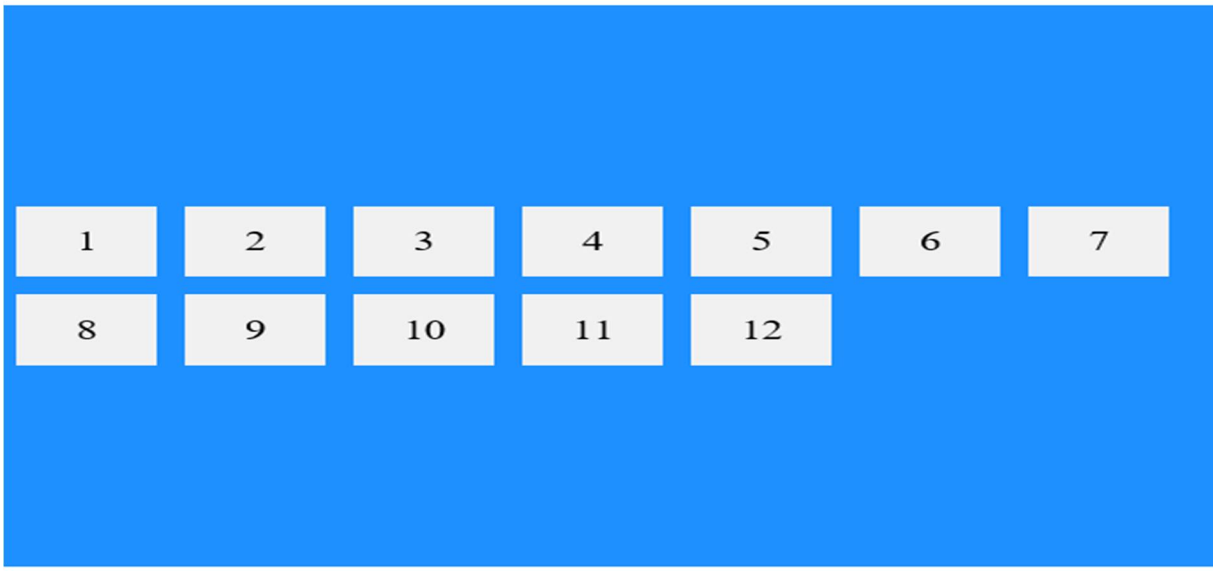
```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
  align-content: space-around;
  background-color: DodgerBlue;
}
```



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

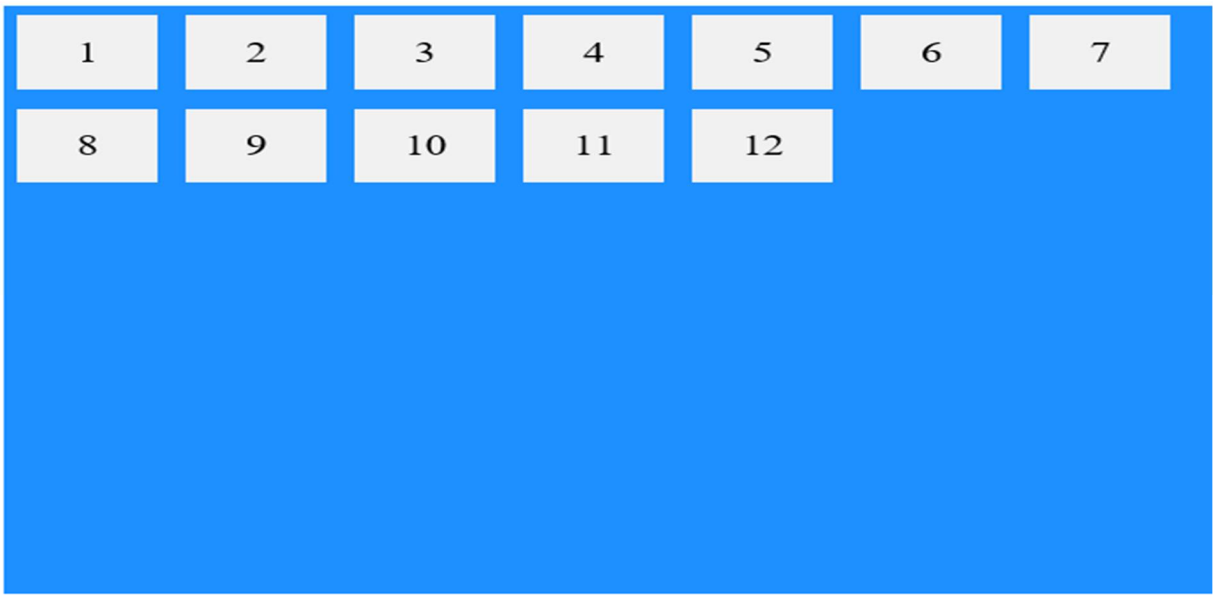
```
.flex-container {
  display: flex;
  height: 600px;
  flex-wrap: wrap;
```

```
  align-content: center;
}
```



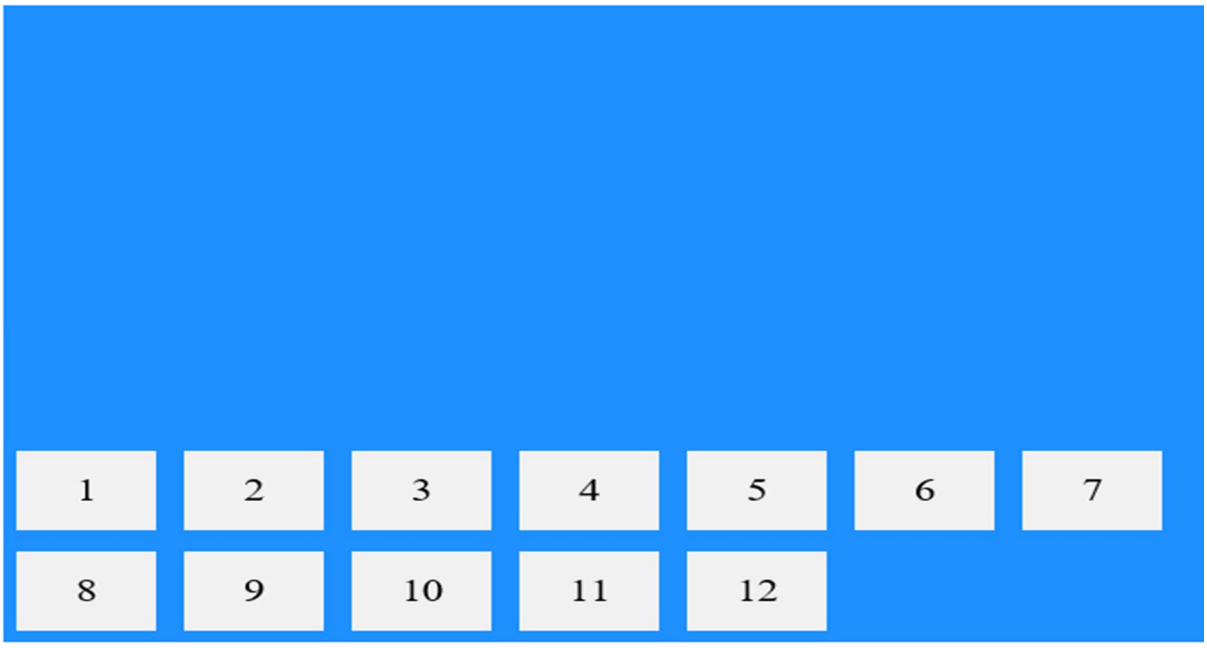
The *flex-start* value displays the flex lines at the start of the container:

```
.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.

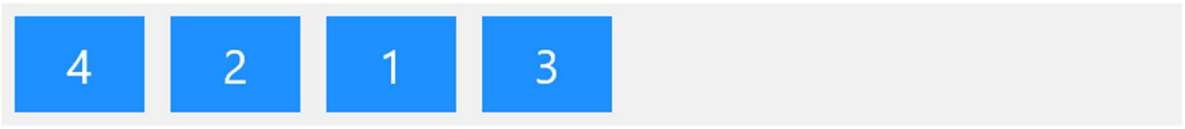
```
.flex-container {
  display: flex;
  height: 300px;
  justify-content:center;     ….x-axis
  align-items: center;        ….y-axis
}
```

……..

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.

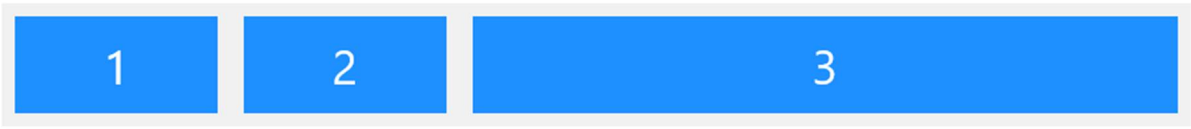
The order value must be a number, default value is 0.

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.

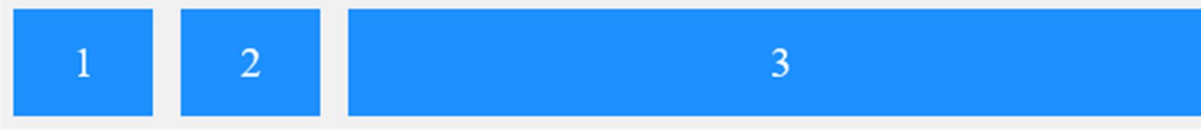
Example

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 لمحاذاةfor the selected item inside the flexible container.

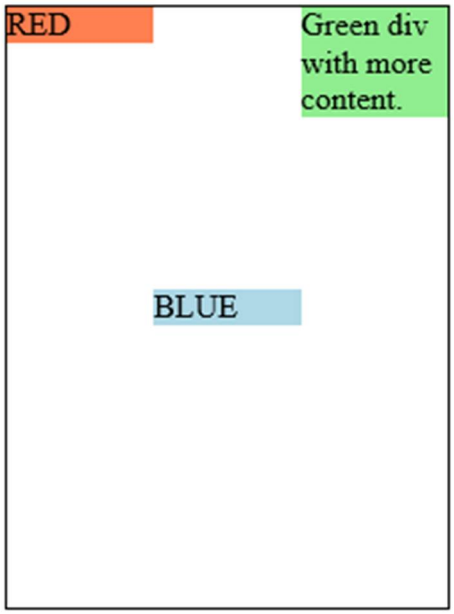
The `align-self` property overridesلغى the default alignment set by the container's `align-items` property.

```
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%;

HTML Input Placeholder

HTML

```
<input type="text" id="id8" placeholder="Here your name">
```

Css

```
#id8::placeholder {
```

```
    color: lightblue;

}
```

Css :root { vs body,nav,li, a, img, p {

```
:root {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

:root will not effect on the items that have default values.

Ex. P has default value, so :root will not effect on P.

So the solution :

```
body, nav, li, a, img, p {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

***CSS Image Sprites**

An image sprite is a collection of images put into a single image.

A web page with many images can take a long time to load and generates multiple server requests.

Using image sprites will reduce the number of server requests and save bandwidth.

SASS SCSS

Sass (Syntactically Awesome Style Sheets) is an extension of CSS that enables you to use things like variables, nested rules, inline imports and more. It also helps to keep things organised and allows you to create style sheets faster.

1. Download node.js from www.nodejs.org site. Have the 8.10 version. The one on the left
2. go to terminal. To do so type cmd in searchbar. Open it and run it as admin.
3. type node -v. If it says the version then you have succesfully installed node.js
4. then you type npm install node-sass -g. Dont forget the -g flag. Important
5. In atom download the package to preferences that is called sass-autcompile.
6. Go to settings and change from output compressed style to expanded style.

Enjoy Sass-magic!

https://sass-lang.com/guide

https://www.sassmeister.com

https://sass-lang.com/documentation/file.SASS_REFERENCE.html

1-Nested Syntax بناء جملة متداخلة

```
nav
{
  ul4
  {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a
  {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

```
nav ul4 {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

2-name spaces , ... I never like it

```
article {
    text:
    {
        align: center;
        direction: none;
    }
}
```

3-partials جزئيات

```
_nav.scss    _form.scss    _footer.scss
```

```
// main.scss
@import 'nav';
@import 'form';
@import 'footer';
```

4-Variables

Important note :

-Use the variable in properties as **$var**

Ex :

```
$text_size: 3vw;

li:nth-of-type(1){ font-size: $text_size;}
```

-Use the variable in Selectors as **#{$var}**

Ex :

```
$ele: electornics;   // ele is varible for the the class 'electornics'

section.#{$ele} {  color: red; }
```

DartSass v1.6.2

```scss
$MHD_color: #00FF82;
$blue_color: #100CE8;

li:nth-last-of-type(1){background:  $MHD_color;}
li:nth-last-of-type(2){background:  $blue_color;}
li:nth-last-of-type(3){background:  $MHD_color;}
li:nth-last-of-type(4){background:  $blue_color;}
li:nth-last-of-type(5){background:  $MHD_color;}
```

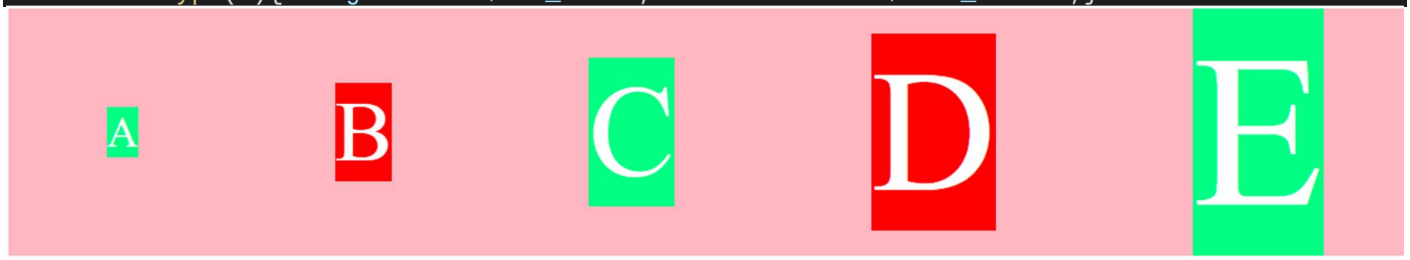
CSS

```css
li:nth-last-of-type(1) {
  background: #00FF82;
}

li:nth-last-of-type(2) {
  background: #100CE8;
}

li:nth-last-of-type(3) {
  background: #00FF82;
}

li:nth-last-of-type(4) {
  background: #100CE8;
}

li:nth-last-of-type(5) {
  background: #00FF82;
}
```

5-Operators

Doing math in your CSS is very helpful. Sass has a handful of standard math operators like `+`, `-`, `*`, `/`, and `%`

```scss
$MHD_color: #00FF82;
$blue_color: #100CE8;
$text_size: 3vw;

li:nth-of-type(1){background:  $MHD_color;    font-size: $text_size;}
li:nth-of-type(2){background:  $blue_color;   font-size: $text_size*2;}
li:nth-of-type(3){background:  $MHD_color;    font-size: $text_size*3;}
li:nth-of-type(4){background:  $blue_color;   font-size: $text_size*4;}
li:nth-of-type(5){background:  $MHD_color;    font-size: $text_size*5;}
```



Or to calculate the width and height

DartSass v1.6.2

```scss

article{

  width: 600px / 960px * 100%;
}

aside {

  width: 300px / 960px * 100%;
}
```

CSS

```css
article {
  width: 62.5%;
}

aside {
  width: 31.25%;
}
```

6- Variable for Class .#{$var} or Variable for ID ##{$var}

HTML

```
<section class="electornics"></section>
```

scss

```
$ele: electornics;   // ele is varible for the the class 'electornics'

section.#{$ele} {  color: red; }
```

will convert to css

```
section.electornics {
  color: red;
}
```

7-for Loop , like ruby @for $i from 1 to 10 { }

scss

```
nav {

  @for $i from 1 to 11 {
    li:nth-of-type(#{$i}) { font-size: 1vw * $i;  background: rgba(48, 49, 255, $i/10);}
  }
```

will convert to css

```
nav li:nth-of-type(1) {
  font-size: 1vw;
  background: rgba(48, 49, 255, 0.1);
}

nav li:nth-of-type(2) {
  font-size: 2vw;
  background: rgba(48, 49, 255, 0.2);
}

nav li:nth-of-type(3) {
  font-size: 3vw;
  background: rgba(48, 49, 255, 0.3);
}

nav li:nth-of-type(4) {
  font-size: 4vw;
  background: rgba(48, 49, 255, 0.4);
}

nav li:nth-of-type(5) {
  font-size: 5vw;
  background: rgba(48, 49, 255, 0.5);
}
```

8- if , like ruby @if $i == 1 { } @else if $i > 1 { } @else if $i > 1 { } @else { }

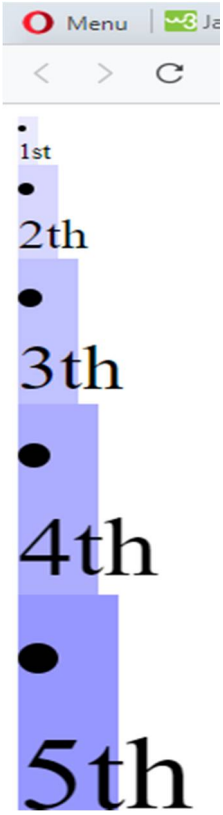
```
@if  $pos == 1 { span:before {content: 'only '+'#{$quantity} ';} }
@else if  $pos == 2 { span:before {content: 'only '+'#{$quantity} ';} }
@else if  $pos == 3 { span:before {content: 'only '+'#{$quantity} ';} }
```

```
@else  { span:before {content: 'only '+'#{$quantity} ';} }
```

```
nav
 {

  @for $i from 1 to 6
   {
    li:nth-of-type(#{$i})
     {
       font-size: 1vw * $i;
       background: rgba(48, 49, 255, $i/10);
       width: $i*10 + px;
       &:after
       {
        @if  $i == 1 {  content:'#{$i}'+ st;    }@else if  $i > 1 {  content:'#{$i}'+ th;
}


       }
     }
    }
}
```

Css

```
nav li:nth-of-type(1) {
  font-size: 1vw;
  background: rgba(48, 49, 255, 0.1);
  width: 10px;
}

nav li:nth-of-type(1):after {
  content: "1st";
}

nav li:nth-of-type(2) {
  font-size: 2vw;
  background: rgba(48, 49, 255, 0.2);
  width: 20px;
}

nav li:nth-of-type(2):after {
  content: "2th";
} ……. an so on
```



7- **each** Loop

The `@each` directive usually has the form `@each $var in <list or map>`. `$var` can be any variable name, like `$length` or `$name`, and `<list or map>` is a SassScript expression that returns a list or a map.

The `@each` rule sets `$var` to each item in the list or map, then outputs the styles it contains using that value of `$var`.

```
@each $animal in puma, sea-slug, egret, salamander {
  .#{$animal}-icon {
    background-image: url('/images/#{$animal}.png');
  }
}
```

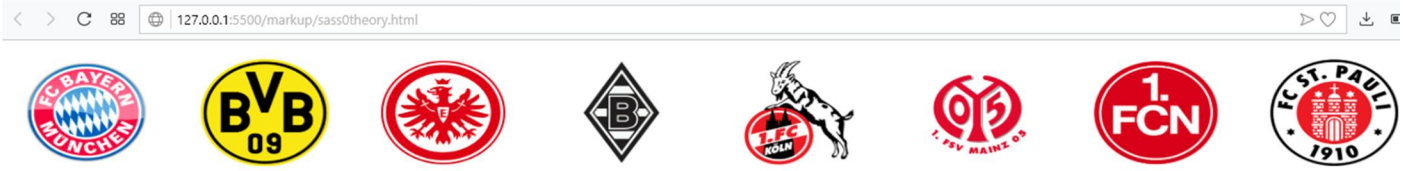
is compiled to:

```
.puma-icon {
  background-image: url('/images/puma.png'); }
.sea-slug-icon {
  background-image: url('/images/sea-slug.png'); }
.egret-icon {
  background-image: url('/images/egret.png'); }
.salamander-icon {
  background-image: url('/images/salamander.png'); }
```

Multiple Assignment

```
@each $animal, $color, $cursor in (puma, black, default),
                                  (sea-slug, blue, pointer),
                                  (egret, white, move) {
  .#{$animal}-icon {
    background-image: url('/images/#{$animal}.png');
    border: 2px solid $color;
    cursor: $cursor;
  }
}
```

```
@each $pos, $team in (1, bayern), (2, dortmund), (3, frankfurt), (4, gladbach), (5, koln),
(6, mainz), (7, nurnberg), (8, pauli)
      {
        &:nth-of-type(#{$pos})
         {
          background-image: url(../images/#{$team}.png);
         }
      }
```



Ex in loop: 14 lines in scss will generate 50 lines in css

Scss

```
nav
 {

  @for $i from 1 to 6
   {
    li:nth-of-type(#{$i})
     {
       font-size: 1vw * $i;
       background: rgba(48, 49, 255, $i/10);
       width: $i*10 + px;
       &:after {  content:'#{$i}'; }
     }
   }
}
```

It will generate the CSS

```css
nav li:nth-of-type(1) {
  font-size: 1vw;
  background: rgba(48, 49, 255, 0.1);
  width: 10px;
}

nav li:nth-of-type(1):after {
  content: "1";
}

nav li:nth-of-type(2) {
  font-size: 2vw;
  background: rgba(48, 49, 255, 0.2);
  width: 20px;
}

nav li:nth-of-type(2):after {
  content: "2";
}
```

```css
nav li:nth-of-type(4):after {
  content: "4";
}

nav li:nth-of-type(5) {
  font-size: 5vw;
  background: rgba(48, 49, 255, 0.5);
  width: 50px;
}

nav li:nth-of-type(5):after {
  content: "5";
}
/*# sourceMappingURL=sass_theory.css.map */
```

Css sectors > + ~

space all children and grandchildren. div li

> all children, no grandchildren. div > li

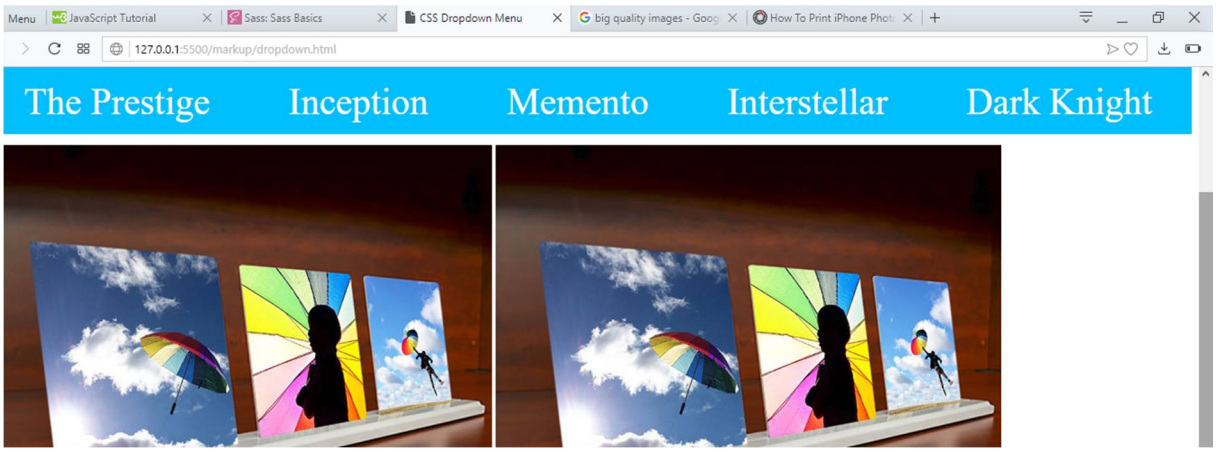
~ all next siblings, not before siblings. div ~ li

+ only the next direct sibling. div + li

css position: sticky;لزج

the user will still see the nav while he scrolls down the page

```css
nav {
  margin: 1.1vw 0;
  background: deepskyblue;
  height: 5.4vw;
  position: sticky;
  top: 0;
}
```



css background-blend-mode مزج Property

this property defines the blending مزج mode of each background layer (color and/or image).

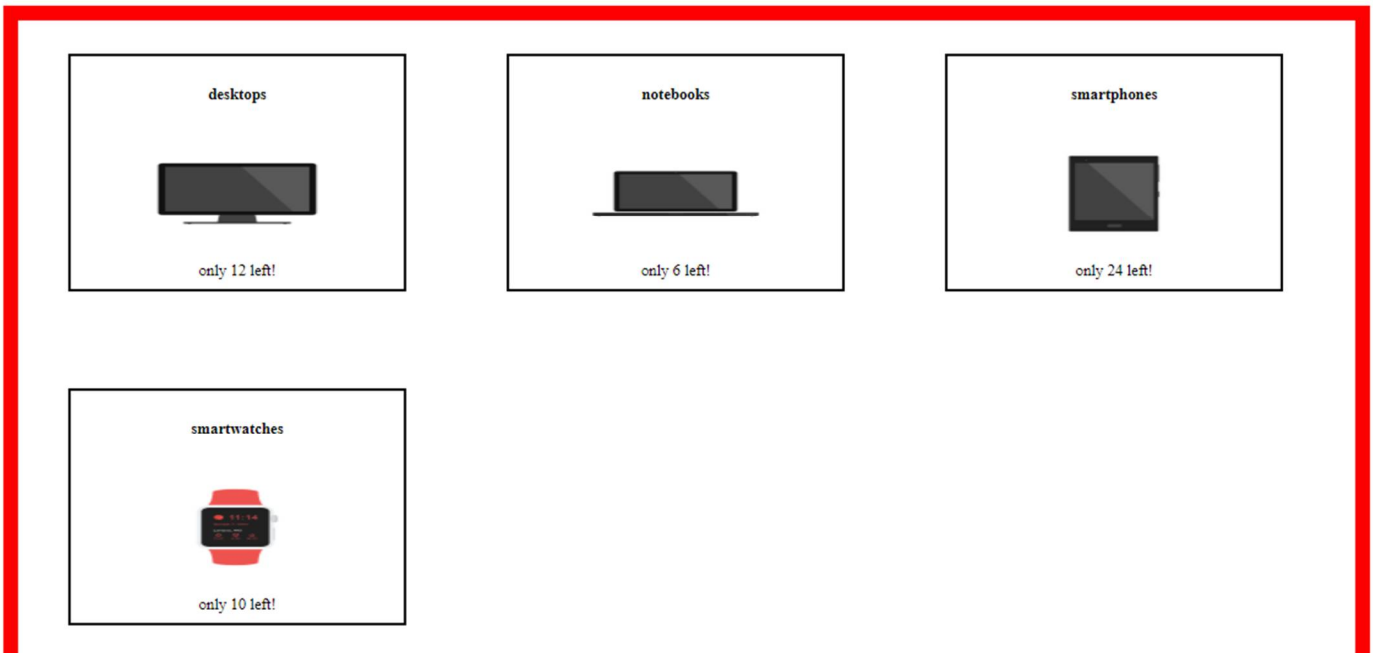
Css Example

```
footer {
  height: 30vw;
  background: url(../../images/planesky.jpg) orange;
  background-repeat: no-repeat;
  background-size: 100% 100%;
  background-blend-mode: darken;
}
```

Scss Media queries, nth-last-of-type

```
nav
{
    display: flex;
    flex-direction: row;

    @media screen and (max-width: 768px) { flex-direction: column; }

    Li
   {
        list-style-type: none;

        @media screen and (max-width: 768px) { font-size: 5vw; }


        &:nth-last-of-type(even) { background: red; }

    }
}
```

Scss Shop Exercise




```
body
 {
  main
  {
    border: 15px red solid;
    display: flex;
    justify-content: flex-start;
    flex-wrap: wrap;
    margin: 10vw auto;
    width: 80vw;
    height: 40vw;
    section {
      margin: 2vw 3vw;
      border: 2px black solid;
      display: flex;
      flex-direction: column;
      justify-content: space-around;
      align-items: center;
      width: 25%;
      height: 35%;
      div {
        padding: 3vw 6vw;
        background-size: 100% 100% ;
        background-repeat: no-repeat;
        background-position: center center;
          }
      @each $pos, $team, $quantity in (1, desktops, 12), (2, notebooks, 6), (3,
smartphones, 24), (4, smartwatches, 10)
      {
        &:nth-of-type(#{$pos})
        {
            div
            {
              background-image: url(../images/#{$team}.png);
```

```
            }
            h4:before
            {
                content: '#{$team}';
            }
            @if  $pos == 1 { span:before {content: 'only '+'#{$quantity} ';} }
            @else if  $pos == 2 { span:before {content: 'only '+'#{$quantity} ';} }
            @else if  $pos == 3 { span:before {content: 'only '+'#{$quantity} ';} }
            @else  { span:before {content: 'only '+'#{$quantity} ';} }
        }
      }
    }
  }
}
```

SASS Placeholder –to call group of public variables (not in css)

The same properties and the same values

To create it :

%Name_group_variables { }

To call him :

@extend %Name_group_variables;

```
%Name_group_variable {
    display: flex;
    flex-direction: column;
}

form, div {
    @extend %Name_group_variable;
}
```

sass Example

```
%set_padding {
    padding:0 vw;
}
%btn_button {
    font-family: thoma;
    font-size: 1.5vw;
    padding: 1vw 2vw;
}
%make_flex {
    display: flex;
    flex-direction: column;
    justify-content: space-around;
    align-items: center,
}
p, h1 {
    @extend %set_padding;
}
```

```
div {
    @extend %make_flex;
}
button {
    @extend %btn_button;
}
```

SASS mixin –to call group of public variables (not in css)

The same properties but the different values

To create it :

@Name_group_variables($var1, $var2, $var3, $var4, $var5: default_value_var5) { }

To call him :

@include ame_group_variables(vlaue, vlaue, vlaue, vlaue, value or nothing to get default);

```
@mixin make_flex($display, $direction, $justify, $align, $width, $height: auto) {
    display: $display;
    direction: $direction;
    justify-items: $justify;
    align-items: $align;
    width: $width;
    height: $height;
}

div {
    @include make_flex(flex, column, space-around, center, 75vw);
}

form {
    @include make_flex(flex, row, center, center, 90vw);
}
```

Sass Script Functions

https://sass-lang.com/documentation/file.SASS_REFERENCE.html#functions

……..

bootstrapstudio

A powerful desktop app for creating responsive websites using the Bootstrap framework.

Bootstrap is the most popular HTML, CSS, and JavaScript framework for developing responsive, mobile-first websites.

Bootstrap is completely free to download and use!

1- Include CDN (online)

copy the css js links to html

https://getbootstrap.com/docs/4.0/getting-started/introduction/

when we set js link in the top of html, is special event that is called (on Dom content load)

2-Download vs Include CDN

Download:1- work offline,2- to see bootstrap content, 3-Modify

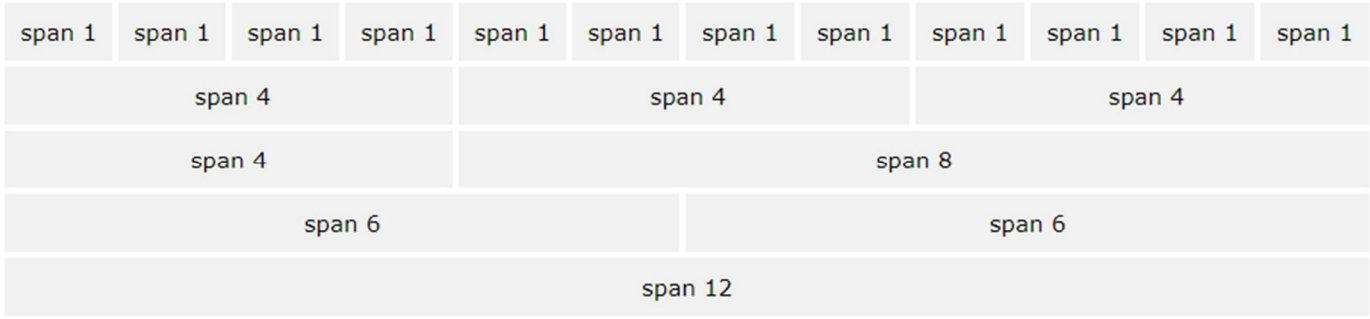
Include CDN: 1- Minified versions, parse تحليل faster, load faster, better production تحضير

bootstrapstudio

Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.

Then the Rows define as stack.

Important note : when I write `class="row"` , then mains that row is “Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.”



Grid Classes

The Bootstrap 4 grid system has five classes:

- `.col-` (extra small devices - screen width less than 576px)
- `.col-sm-` (small devices - screen width equal to or greater than 576px)
- `.col-md-` (medium devices - screen width equal to or greater than 768px)
- `.col-lg-` (large devices - screen width equal to or greater than 992px)
- `.col-xl-` (xlarge devices - screen width equal to or greater than 1200px)

The classes above can be combined to create more dynamic and flexible layouts.

Tip: Each class scales up, so if you wish to set the same widths for `sm` and `md`, you only need to specify `sm`.

create a row (`<div class="row">`). Then, add the desired number of columns (tags with appropriate `.col-*-*` classes). The first star (*) represents the responsiveness: sm, md, lg or xl, while the second star represents a number, which should add up to 12 for each row.

instead of adding a number to each `col`, let bootstrap handle the layout, to create equal width columns: two `"col"` elements = 50% width to each col. three cols = 33.33% width to each col. four cols = 25% width, etc. You can also use `.col-sm|md|lg|xl` to make the columns responsive.

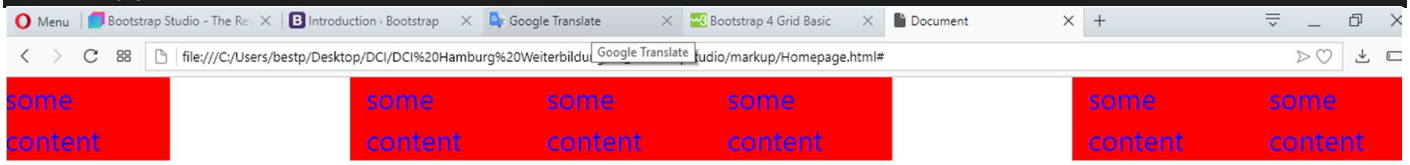
Explain from teacher:

```
<div class="row">      …class row means to set a new row
      <div class="example col-4">some content</div>
```

```
      <div class="col-1"></div>
      <div class="example col-4">some content</div>
      <div class="example col-2">some content</div> class col-2 means to span a 2 cols
</div>
```

```
.example {
      background: red;
      color: blue;
      font-size: 2vw;
}
```


```
      <div class="row">
            <div class="example col">some content</div>
            <div class="col"></div>
            <div class="example col">some content</div>
            <div class="example col">some content</div>
      </div>

      <div class="row">
            <div class="example col">some content</div>
            <div class="col"></div>
            <div class="example col">some content</div>
            <div class="example col">some content</div>
      </div>
```



Text Colors

Bootstrap 4 has some contextual classes that can be used to provide "meaning through colors".

The classes for text colors are: `.text-muted`, `.text-primary`, `.text-success`, `.text-info`, `.text-warning`, `.text-danger`, `.text-secondary`, `.text-white`, `.text-dark`, `.text-body` (default body color/often black) and `.text-light`:

Example

This text is muted.

This text is important.

This text indicates success.

This text represents some information.

This text represents a warning.

This text represents danger.

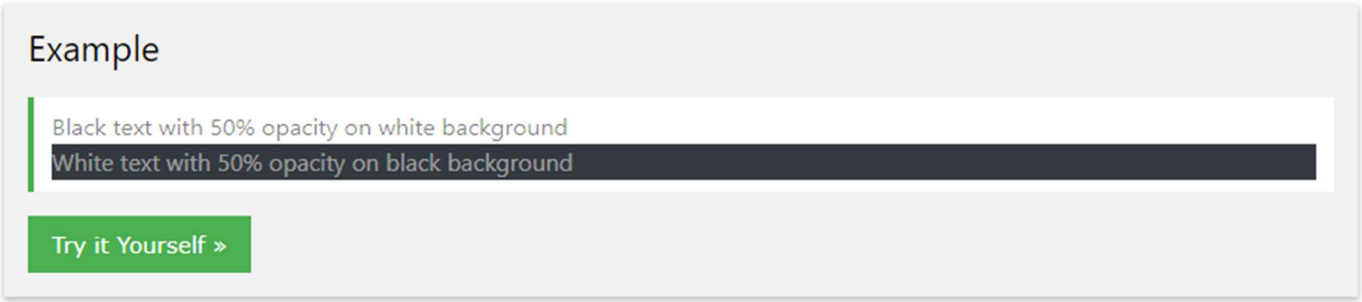
Secondary text.

Dark grey text.

Body text.

```
<div class="container">
  <h2>Contextual Colors</h2>
  <p>Use the contextual classes to provide "meaning through colors":</p>
  <p class="text-muted">This text is muted.</p>
  <p class="text-primary">This text is important.</p>
  <p class="text-success">This text indicates success.</p>
  <p class="text-info">This text represents some information.</p>
  <p class="text-warning">This text represents a warning.</p>
  <p class="text-danger">This text represents danger.</p>
  <p class="text-secondary">Secondary text.</p>
  <p class="text-dark">This text is dark grey.</p>
  <p class="text-body">Default body color (often black).</p>
  <p class="text-light">This text is light grey (on white background).</p>
  <p class="text-white">This text is white (on white background).</p>
</div>
```

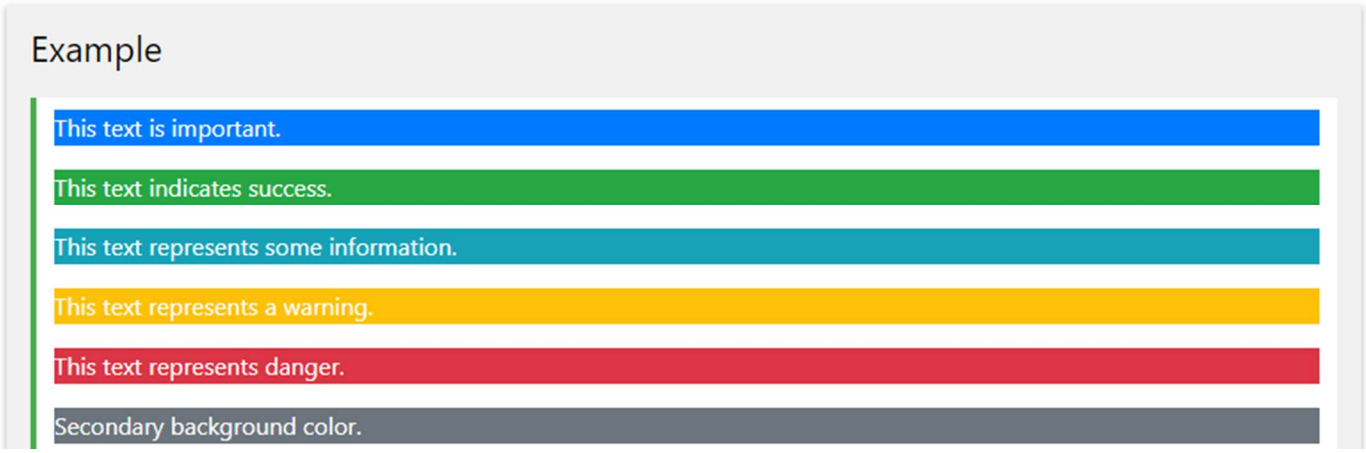
You can also add 50% opacity for black or white text with the `.text-black-50` or `.text-white-50` classes:



Background Colors

The classes for background colors are: `.bg-primary`, `.bg-success`, `.bg-info`, `.bg-warning`, `.bg-danger`, `.bg-secondary`, `.bg-dark` and `.bg-light`.

Note that background colors do not set the text color, so in some cases you'll want to use them together with a `.text-*` class.



Class container vs container-fluid

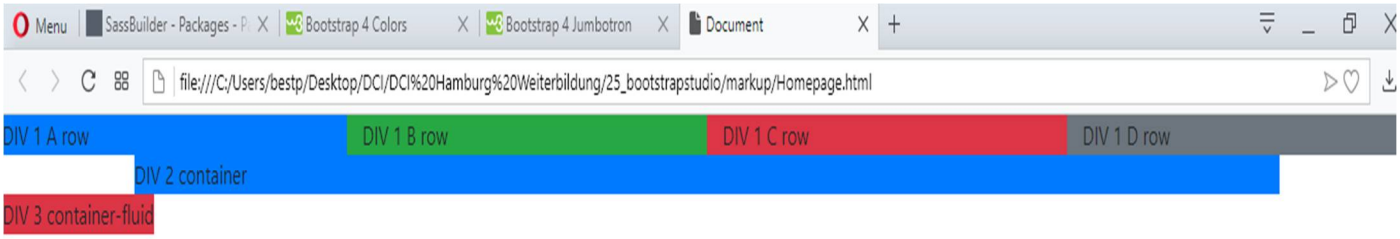
Container class: the element will take the width of 80% and will set in center.

Container-fluid class: the element will take the width of 100%.

```
<body>

    <div class="row" >
```

```
        <div class="col-12  col-md-6 col-lg bg-primary">DIV 1 A row</div>
        <div class="col-12  col-md-6 col-lg bg-success">DIV 1 B row</div>
        <div class="col-12  col-md-6 col-lg bg-danger">DIV 1 C row</div>
        <div class="col-12  col-md-6 col-lg bg-secondary">DIV 1 D row</div>
    </div>

    <div class="container" >
        <div class=" bg-primary">DIV 2 container</div>
    </div>

    <div class="row container-fluid" >
        <div class=" bg-danger">DIV 3 container-fluid</div>
    </div>

</body>
```

https://getbootstrap.com https://getbootstrap.com



offline bootstrap

https://getbootstrap.com

bootstrap The align-items Property : ***stretch vs Flex-start***

Css Example

----------finish bootstrap.....

emmet html5 shortcut

https://docs.emmet.io/cheat-sheet/

note: make sure where is your mouse courser, it muss to be in the end

css Example

Css Example

css Example

Css Example

css Example

Css Example

css Example

Css Example

css Example

Css Example

CSS LINK

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