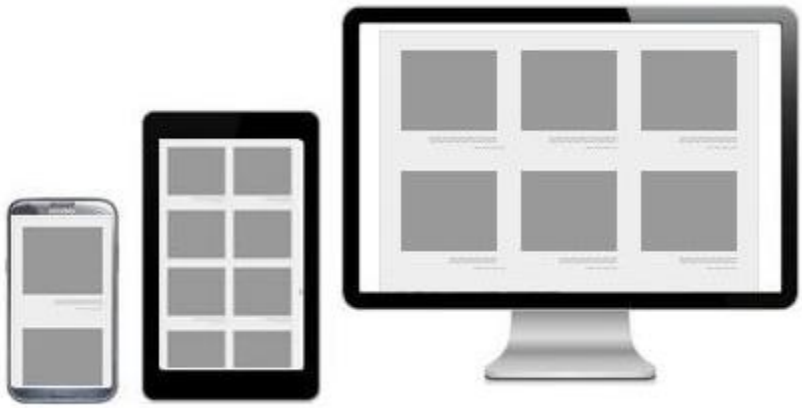
CSS – Grid System

In grid system the first thing we read about is a container. It a special class which Bootstrap provides.

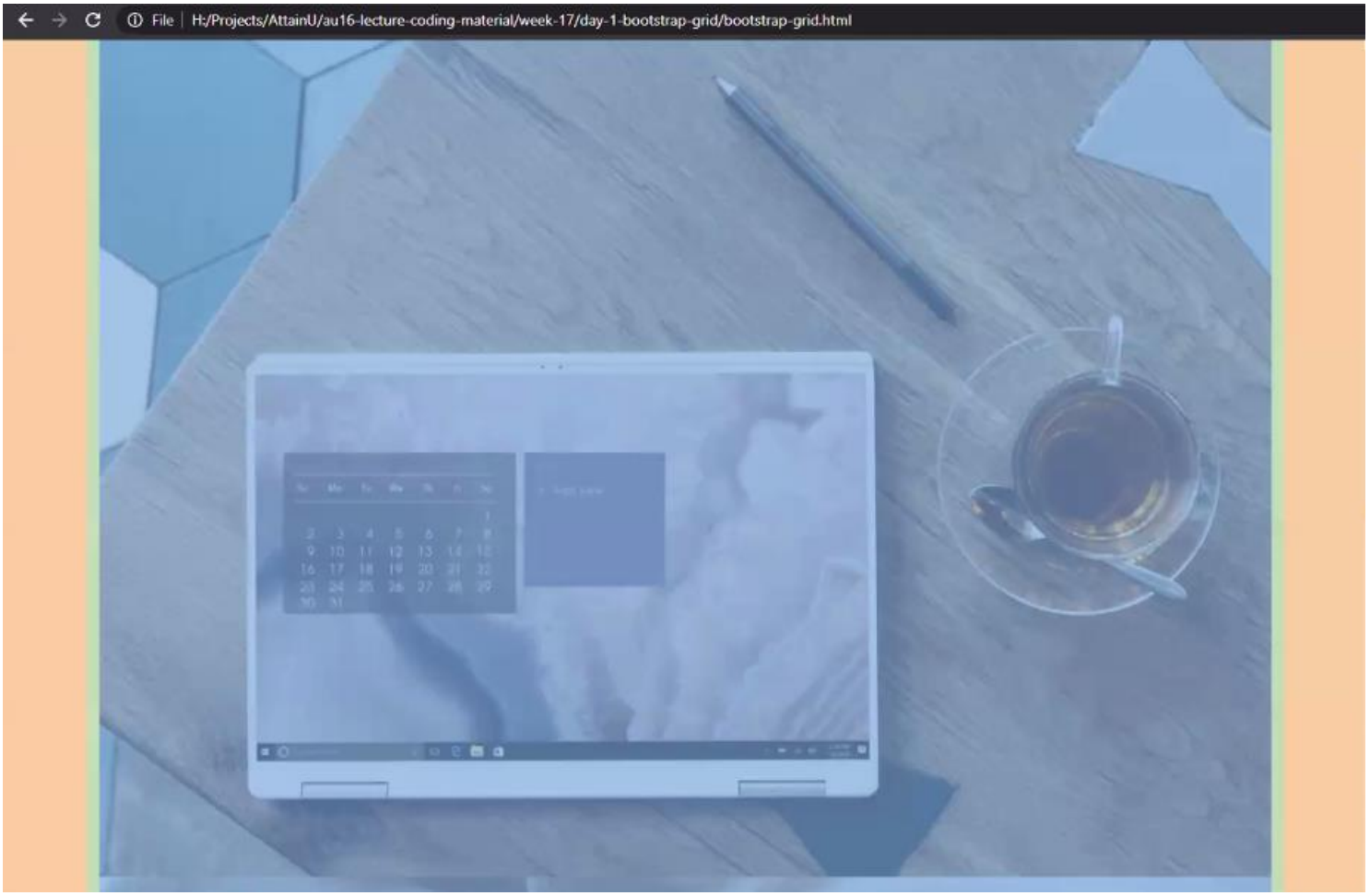
.img-fluid → makes image responsive.

Responsive images will automatically adjust to fit the size of the screen. All the images which have the class="unglued" will resize themselves based on the width of the parent element.



Ex: 01

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



TO have 3 columns,

The images are spanning the entire 100% width of the browser. But we want all the 3 images to the next to each other (Instagram look). So before we start that we will put an div element around the 3 images using a class container. This container class has its own margin. Once we refresh the page, we can see the margin between the image and the browser window.

Media Query (@media): this is just like an if-else statement. In the present webpage, if the viewport has min-width of 992px, then the max-wdith is supposed to be around 960px. Furthermore, if the min-width is 768px, then the max-width should be 720px. And if the min-width is 576px, then the max-width should be 540px.

We see that there is a margin provided on the left and right. if we decrease the browser width, then the size of the image is also being resized. But after the image has been increased to a particular width, the image size does not increase. This is because the class container has given a max-width of 960px.

Anything less than 576px of width (min-width), the container class would now change its max-width to 540px. In other words, if the screen size of less than 576px, the max-width is to be 540px.

This sudden change of max-width happens due to break-points or media query.

Breakpoint:

Core concepts

- Breakpoints are the building blocks of responsive design. Use them to control when your layout can be adapted at a particular viewport or device size.
- Use media queries to architect your CSS by breakpoint. Media queries are a feature of CSS that allow you to conditionally apply styles based on a set of browser and operating system parameters. We most commonly use min-width in our media queries.
- Mobile first, responsive design is the goal. Bootstrap’s CSS aims to apply the bare minimum of styles to make a layout work at the smallest breakpoint, and then layers on styles to adjust that design for larger devices. This optimizes your CSS, improves rendering time, and provides a great experience for your visitors.

Available breakpoints

Bootstrap includes six default breakpoints, sometimes referred to as grid tiers, for building responsively. These breakpoints can be customized if you're using our source Sass files.

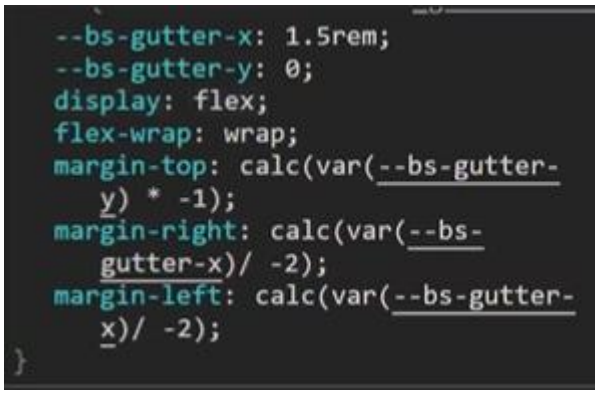
Breakpoint	***Class infix***	***Dimensions***
X-Small	*None*	<576px
Small	sm	≥576px
Medium	md	≥768px
Large	lg	≥992px
Extra large	xl	≥1200px
Extra extra large	xxl	≥1400px

Each breakpoint was chosen to comfortably hold containers whose widths are multiples of 12. Breakpoints are also representative of a subset of common device sizes and viewport dimensions—they don't specifically target every use case or device. Instead, the ranges provide a strong and consistent foundation to build on for nearly any device.

Now, to get the 3 columns view, we will use the class row from bootstrap.

```
<div class="row">
    Test
</div>
```

The following properties have been added to the div element from the class ***container***.



Putting aside **gutter**, we understand the **display: flex, wrap, margin-top, right and left.**

```
<section class="container">
  <div class="row">
     <div class="col">
    <img scr="" class="img-fluid"
     </div>
     <div class="col">
    <img scr="" class="img-fluid"
     </div>
     <div class="col">
    <img scr="" class="img-fluid"
     </div>
  </div>
</section>
```

OUTPUT:

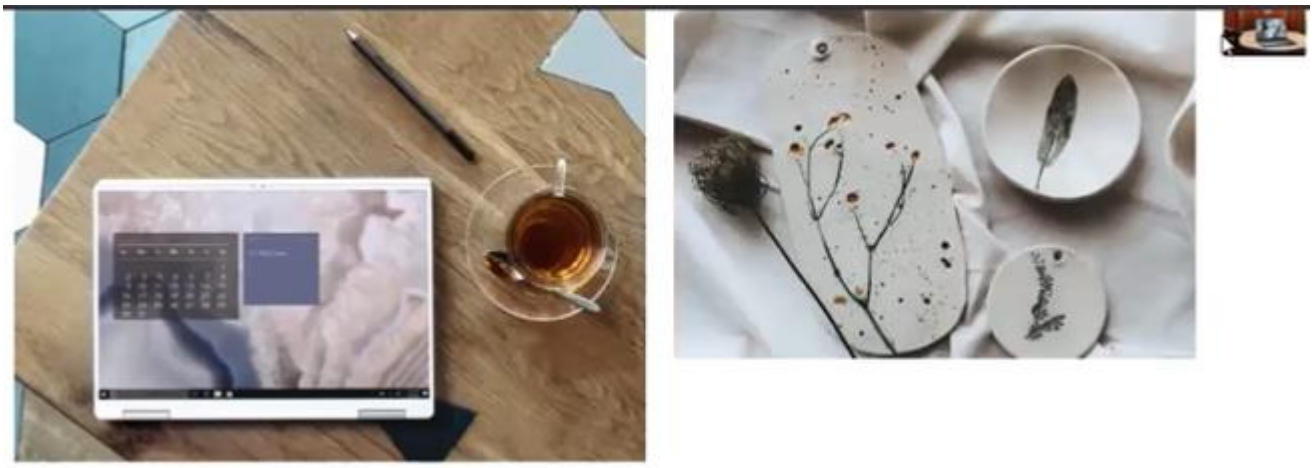
If we have ***n*** number of images, as we are using flex box, all the images would be placed next to each other or in a row. At max Bootstrap allows 12 columns in one row.

Of the 12 columns provided in one row, we need to use equal width and distribute them between the 3 elements. We use the **`class="col"`**. Just like in flexbox, which would provide space equally between its elements, there is another class which enables us to increase size of certain elements.

Let us assign the first element has a column span of 6 out of the 12 column spans available.

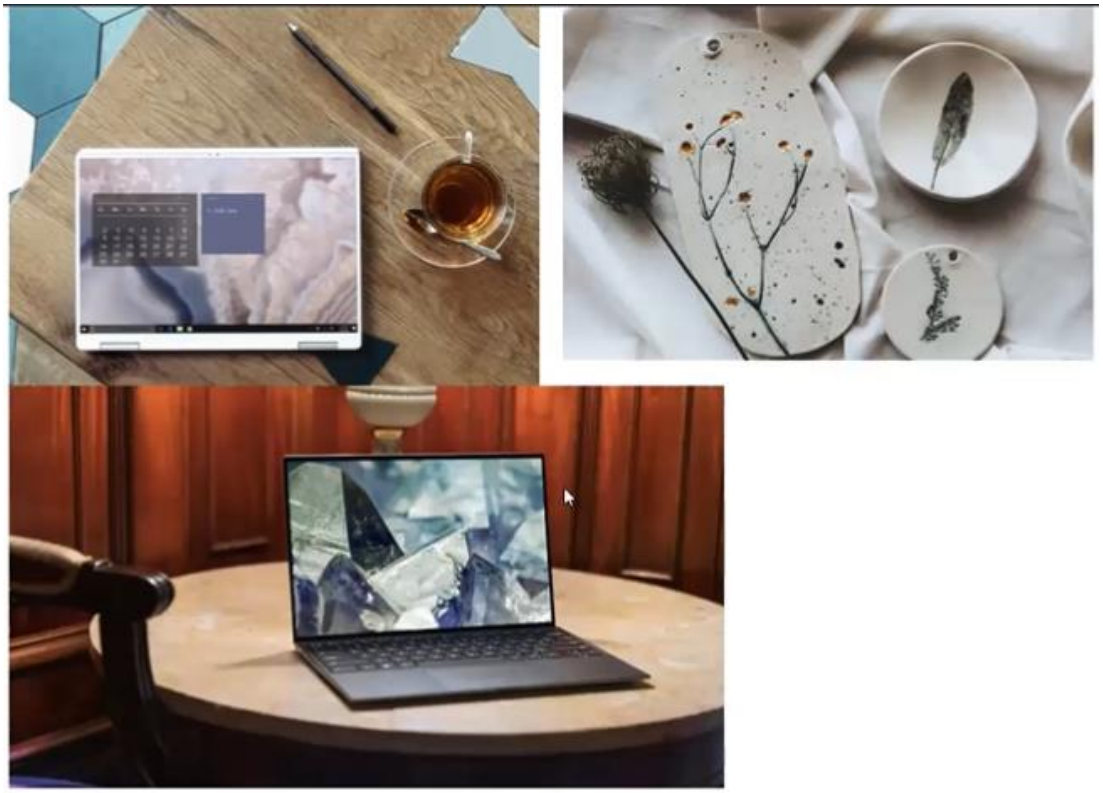
```
<section class="container">
  <div class="row">
     <div class="col-6">
    <img scr="" class="img-fluid"
     </div>
     <div class="col">
    <img scr="" class="img-fluid"
     </div>
     <div class="col">
    <img scr="" class="img-fluid"
     </div>
  </div>
</section>
```

```
<section class="container">
  <div class="row">
     <div class="col-6">
    <img scr="" class="img-fluid"
     </div>
     <div class="col">
    <img scr="" class="img-fluid"
     </div>
     <div class="col-1">
    <img scr="" class="img-fluid"
     </div>
  </div>
</section>
```

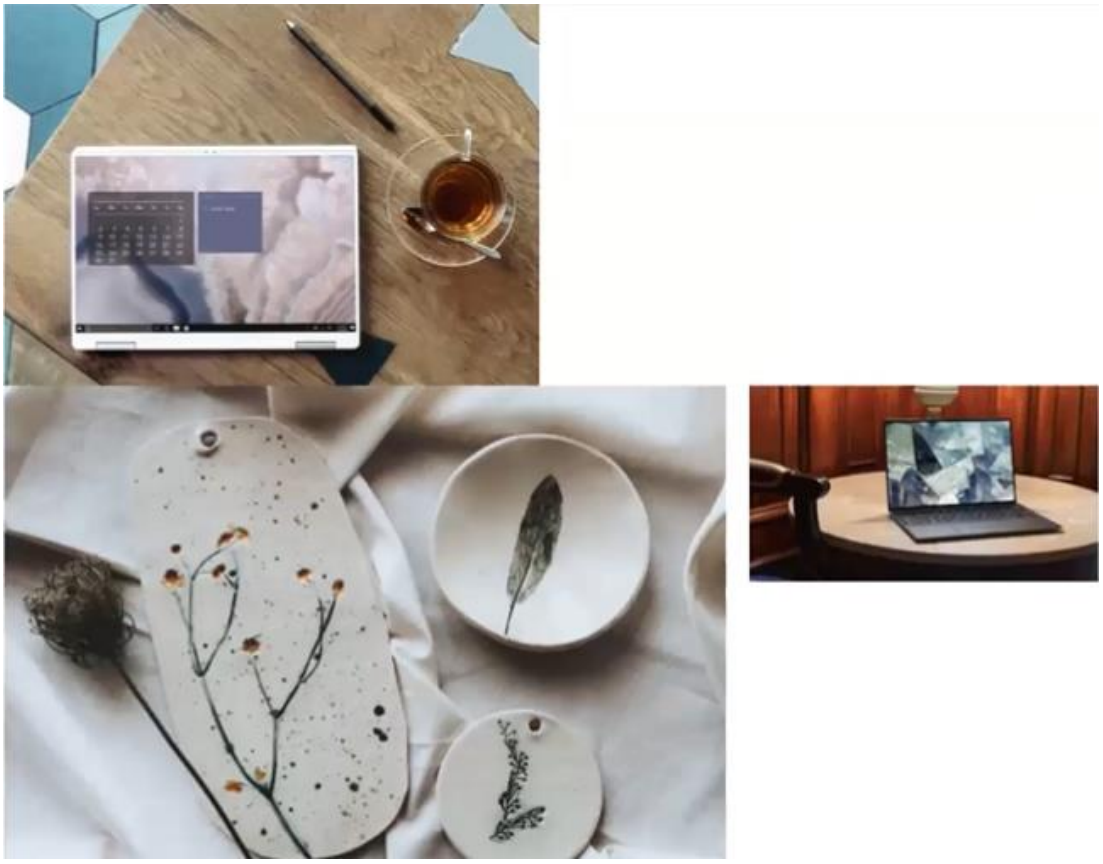


What will happen if we the column span is more than 12.

```
<section class="container">
  <div class="row">
     <div class="col-6">
     <img scr="" class="img-fluid"
     </div>
     <div class="col">
     <img scr="" class="img-fluid"
     </div>
     <div class="col-8">
     <img scr="" class="img-fluid"
     </div>
  </div>
</section>
```



As it cannot accommodate col-8 and col-6 in the same row, bootstrap snaps the element to the next line. This similar to the flex-wrap behaviour. If we add col-8 to the second element, then the second element will snap to the next line not creating any new row. And if the col of the third element is equal or less than the remaining column span, then it will be displayed next to the second element.



If any element has more col span more than 12 then it will snap to the next row (visually only and will not add a new line). Our goal is to have all three columns in one row.

Now, we need to get the column view in the mobile and row view on the desktop.

There is no padding or margin on the top. Instead of using margin or padding for each column, it is good to use the cascading process. If we put **`mt-2`** in our image tag, then we have to repeat it for every image. The same happens for the ***`col`*** and ***`row`*** classes. But if we put it in the class `container` then it will provide margin from the top to all the elements in the container.

All the viewport size corresponds with the list of class infix in the “available breakpoints”. In small view, we want all the images in one column and at a particular break point, we want the images to be in aligned in row view.

Bootstrap is mobile first; it means, we design the website for a mobile view and then design it for the desktop view.

In the mobile view we want all the elements to be in column. Therefore, we can use `col-12` for all elements. This is a basic approach.

All the images are stitched to each other in the mobile view. In order to fix this, we will put the margin-top (mt-2) & margin-bottom (mb-2) to the images so that there is gap between each image. Or else we can use my-2 (vertical axis; top & bottom). But we have to repeat this for all the images.

Gutters:

The white space between two elements is called the gutter. *Gutters* or *alleys* between grid cells can be created using the column-gap and row-gap properties, or the shorthand gap.

We can find the gutter values in Layout details of the bootstrap document. Gutter is for row and we won't be adding it in the column. We will add gx-5 in the row class. We will see more space between columns.

Since we are looking for horizontal gutter, we use gy-5 in the class. And it will provide gap between images in the y-axis. When the browser window is at a particular place, then we need to have a certain width to the elements.

For column, we need to take 4 units of each element to fill the 12 units. So, **`col-md-4`** class is used; meaning, for all the screen size which are less than **`medium break point` (768px)**, then **`col`** size would be **`col-12`** and for all screen size which are more than **`medium break point`**, then col size would be ***`col-4`***.

```
<section class="container">
  <div class="row gy-5">
     <div class="col-12 col-md-4">
     <img scr="" class="img-fluid"
     </div>
     <div class="col-12 col-md-4">
     <img scr="" class="img-fluid"
     </div>
     <div class="col-12 col-md-4">
     <img scr="" class="img-fluid"
     </div>
  </div>
</section>
```



```
<section class="container">
  <div class="row gy-5">
     <div class="col-12 col-md-4 col-xl-6">
     <img scr="" class="img-fluid"
     </div>
     <div class="col-12 col-md-4 col-xl-3">
     <img scr="" class="img-fluid"
     </div>
     <div class="col-12 col-md-4 col-xl-3">
     <img scr="" class="img-fluid"
     </div>
  </div>
</section>
```

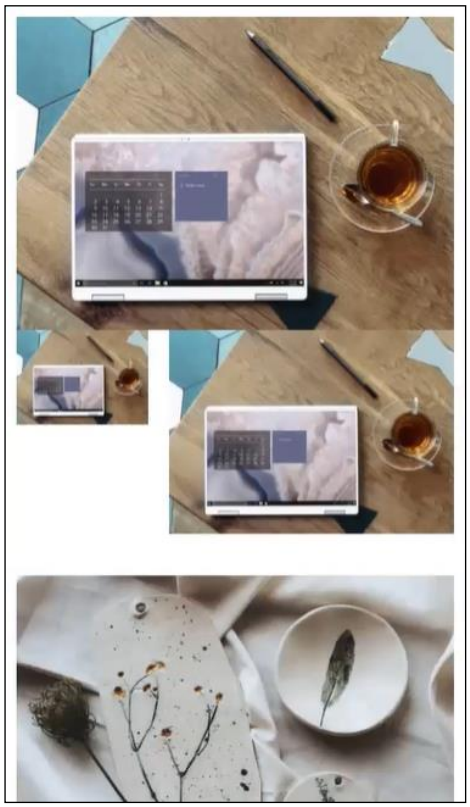
As the web site size increases the view would directly move to row with 3 columns.

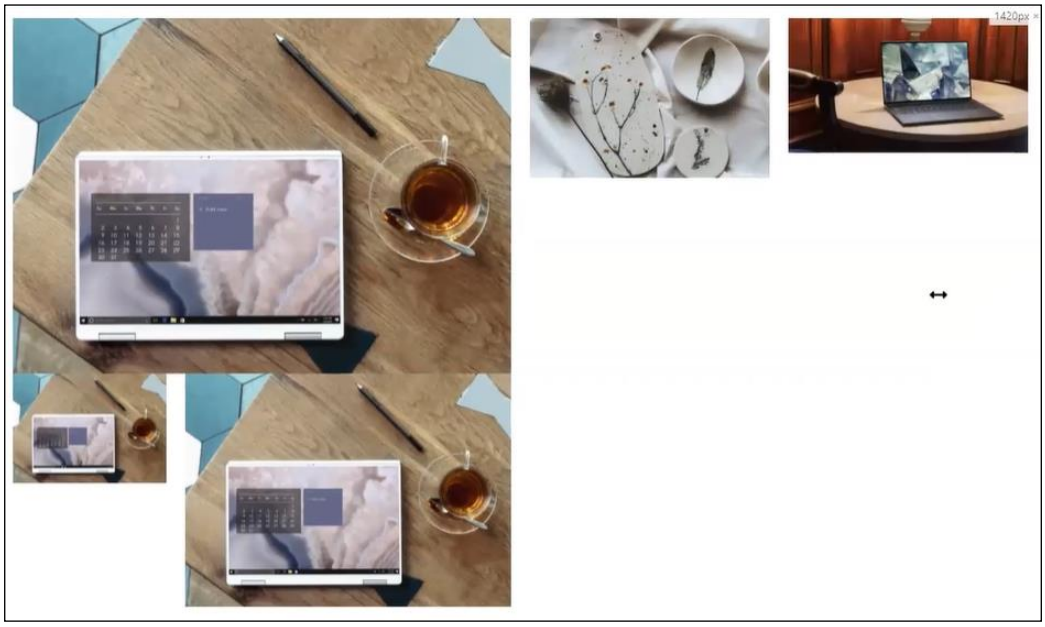
For first image, col-xl-6 is provided and for the remining two, col-xl-3 is provided. So as the max-width of the page goes beyond 1200px, the first element will become twice as large as other two.

Ex: div.row>div.col-4+.col-8 → short hand

```
<section class="row">
  <div class="col-4">
     <img scr="">
  </div>
  <div class="col-8">
     <img scr="">
  </div>
</section>
```

We have created another row and inside it, we have allocated col-4 for the first element and col-8 for the second element.





```
<section class="row">
  <div class="col-4 col-xl-6">
     <img scr="">
  </div>
  <div class="col-8 col-xl-6">
     <img scr="">
  </div>
</section>
```

The nested row has two columns, and once the width of the web page is beyond 1200px, then the two images in the new row would have the same size width.

https://drive.google.com/drive/folders/1ldnNru9kxA6ji95TSQwz_lXZyCnVU3uL

https://drive.google.com/drive/folders/1Gt1gAnwmYzKkth7b7tHXaCLHDGwOhJiR