upGrad

#RahoAmbitious

Full Stack Software Development

upGrad

Course: Introduction to Web Development

Lecture On: CSS Flexbox and CSS Grid

Instructor: Siddhesh Prabhugaonkar

In the previous class, we covered…

- How to hide/show elements
- How to position elements
- How to make elements transparent
- How to apply styles to children and sibling elements of an element?
- How to apply styles only to the first letter or the first word of a text element?

Poll 1 (15 Sec)

Which of the following CSS pseudo-class selector is used to select an element with no children?

1. :nochild
2. :no-child
3. :empty
4. :inherit

Poll 1 (Answer)

Which of the following CSS pseudo-class selector is used to select an element with no children?

1. :nochild
2. :no-child
3. **:empty**
4. :inherit

Poll 2 (15 Sec)

What is the difference between pseudo-classes and pseudo-elements?
(Note: More than one option may be correct.)

1. Pseudo-classes let you style an element, whereas pseudo-elements do not allow it.
2. Pseudo elements can be used to style specific parts of an element whereas pseudo classes can style an element as a whole.
3. Pseudo-classes are represented by a single colon (:), whereas pseudo elements are represented by a double colon (::).
4. None of the above

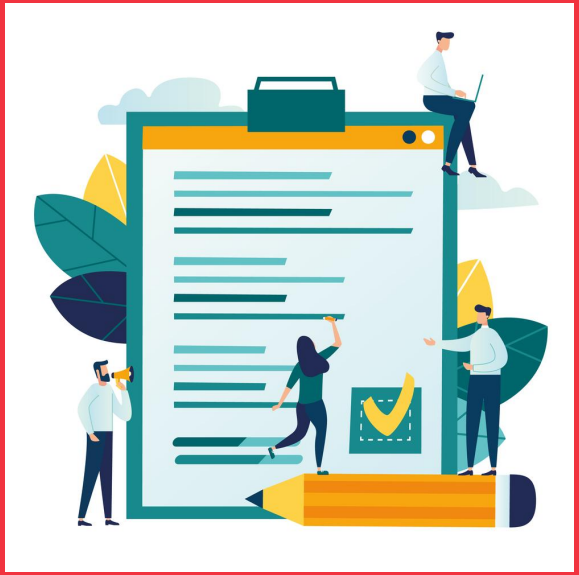
Poll 2 (Answer)

What is the difference between pseudo-classes and pseudo-elements?
(Note: More than one option may be correct.)

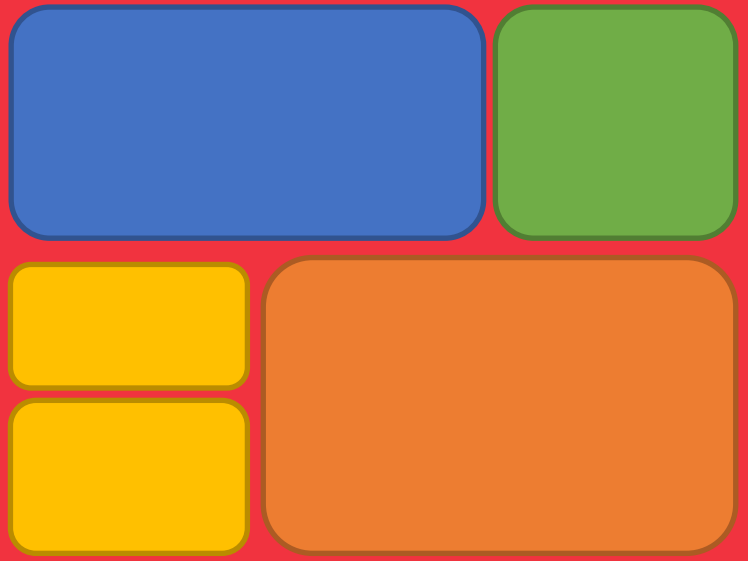
1. Pseudo-classes let you style an element, whereas pseudo-elements do not allow it.
2. **Pseudo elements can be used to style specific parts of an element whereas pseudo classes can style an element as a whole.**
3. **Pseudo-classes are represented by a single colon (:), whereas pseudo elements are represented by a double colon (::).**
4. None of the above

Today's Agenda

- What is flexbox and why is it the most sought-after CSS property?
- How to place elements in a grid using CSS?



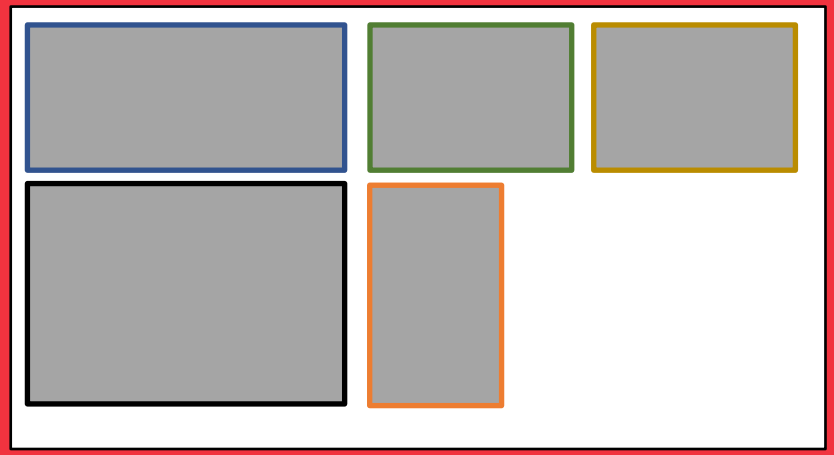
CSS Flexbox



Introduction to Flexbox

- Flexbox stands for **Flexible Box** and aims at providing a flexible way of aligning elements in a container even when their size is unknown. It can distribute space among items in a container.
- The primary principle of a Flexbox is to alter the height/width of an element to best fill the available space in the container. This way, flexbox containers are responsive, and hence, they can be used for adjusting elements on every type of device.
- A flex container either expands items to fill the available space or shrinks them to prevent an overflow.
- A container can be converted to a flexbox by simply applying the CSS property **display: flex** on it. The container then becomes the flex parent/container, whereas the elements inside it become the flex children/items.
- One of the most important advantage of using flexbox is that it can work in both directions, horizontal and vertical, unlike *float*, which only works in the horizontal direction.

Flexbox Parent / Container



Flexbox Parent / Container

Flexbox Parent Properties: flex-direction

- The **flex-direction** property basically defines the direction (normal or reversed) in which the flex children should be placed w.r.t. the main axis. It may be *row* (default), *row-reverse*, *column* or *column-reverse*.

```
<div class="container">
   <div>1</div>
   <div>2</div>
   <div>3</div>
   <div>4</div>
</div>
```

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


```
.container {
   flex-direction: row;
}
```


```
.container {
   flex-direction:
row-reverse;
}
```


```
.container {
   flex-direction: column;
}
```


```
.container {
   flex-direction:
column-reverse;
}
```

Flexbox Parent Properties: flex-wrap

- The **flex-wrap** property defines whether flex items should move down to the next line (wrapping) or adjust their widths and heights to fit in a single row.

```
<div class="container">
   <div>1</div>
   <div>2</div>
   <div>3</div>
   <div>4</div>
   <div>5</div>
   <div>6</div>
</div>
```

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


```
.container {
   flex-wrap: nowrap;
}
```


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

CSS Flexbox

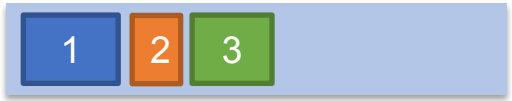
upGrad

Flexbox Parent Properties: justify-content

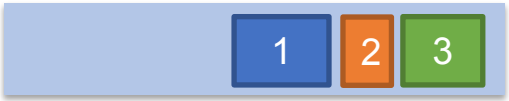
The **justify-content** property defines the alignment of flex items over the main axis. It also distributes the space between & around the items.

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

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




```
.container{ justify-content:
flex-start;}
```

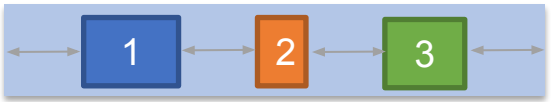



```
.container{ justify-content:
flex-end;}
```


```
.container{ justify-content:
center;}
```


```
.container{ justify-content:
space-between;}
```


```
.container{ justify-content:
space-around;}
```




```
.container{ justify-content:
space-evenly;}
```

Flexbox Parent Properties: align-items

The **align-items** property defines the alignment of flex children/items over the axis perpendicular to the main axis. For instance, if the items are aligned horizontally along the main axis, then the **align-items** property defines the way in which they will be aligned in the vertical space.

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

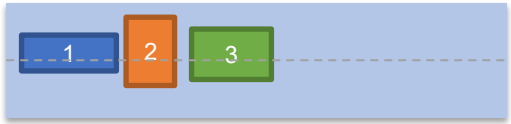
```
.container {
   display: flex;
}
```


```
.container{ align-items:
flex-start;}
```


```
.container{ align-items:
flex-end;}
```


```
.container{align-items:
center;}
```

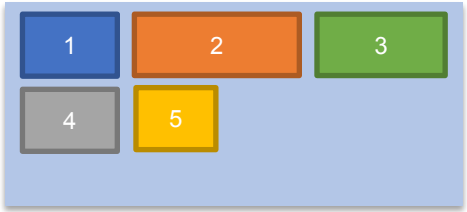

```
.container{ align-items:
stretch;}
```



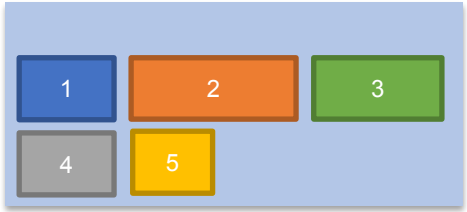

```
.container{ align-items:
baseline;}
```

Flexbox Parent Properties: align-content

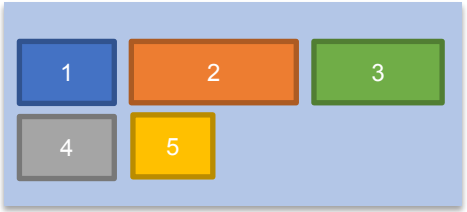
The **align-content** property defines the alignment of flex lines over the axis that is perpendicular to the main axis. Remember that the **content** signifies all the elements, and **items** signify the individual element inside a flex container.



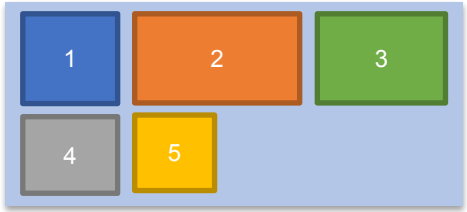

```
align-content: flex-start;
```



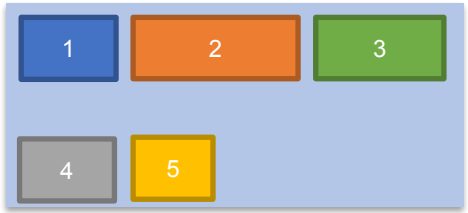

```
align-content: flex-end;
```



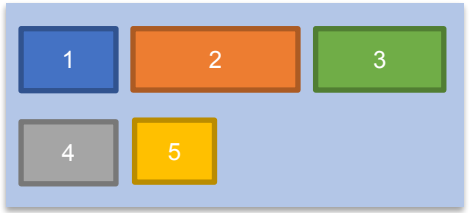

```
align-content: center;
```




```
align-content: stretch;
```




```
align-content:
space-between;
```




```
align-content:
space-around;
```

Poll 3 (15 Sec)

Which of the following is not a property of the CSS flexbox?

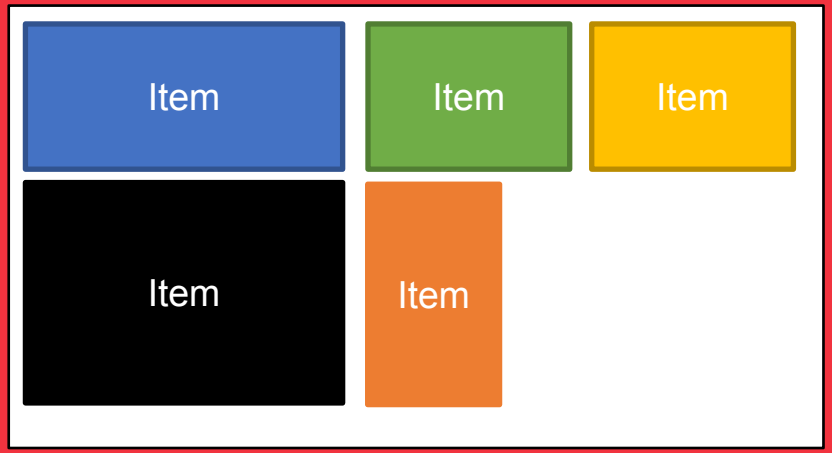
1. flex-direction
2. flex-display
3. justify-content
4. flex-wrap

Poll 3 (Answer)

Which of the following is not a property of the CSS flexbox?

1. flex-direction
2. **flex-display**
3. justify-content
4. flex-wrap

Flexbox Child / Item



Flexbox Child Properties: order

The order property defines the position order of an element in a flexbox container.

```
<div class="container">
   <div id="box1">1</div>
   <div id="box2">2</div>
   <div id="box3">3</div>
   <div id="box4">4</div>
</div>
```

```
.container { display: flex; }

#box1 { order: 3; }
#box2 { order: 4; }
#box3 { order: 1; }
#box4 { order: 2; }
```

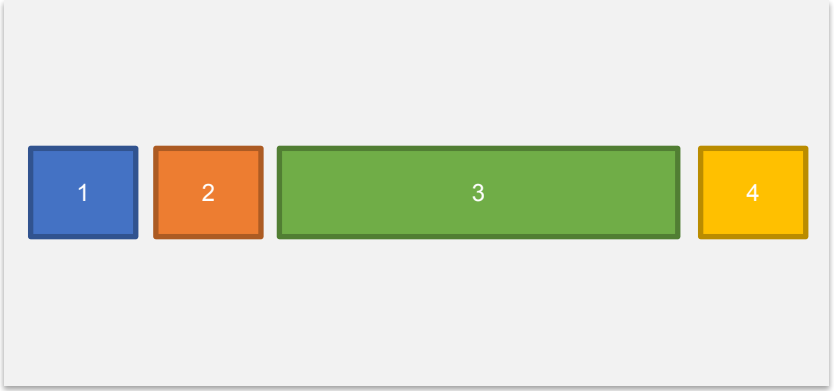
Flexbox Child Properties: flex-grow

The **flex-grow** property specifies the extent to which an item will grow in relation to the other elements inside the flexbox container. It accepts a unitless value for proportion. If all the items have *flex-grow: 1*, then the space will be equally distributed among all the items. For instance, in the example below, box3 will occupy three times the space occupied by the rest.

```
<div class="container">
   <div id="box1">1</div>
   <div id="box2">2</div>
   <div id="box3">3</div>
   <div id="box4">4</div>
</div>
```

```
.container {
   display: flex;
   width: 350px;
}

.container div { width: 50px; }
#box3 { flex-grow: 3; }
```



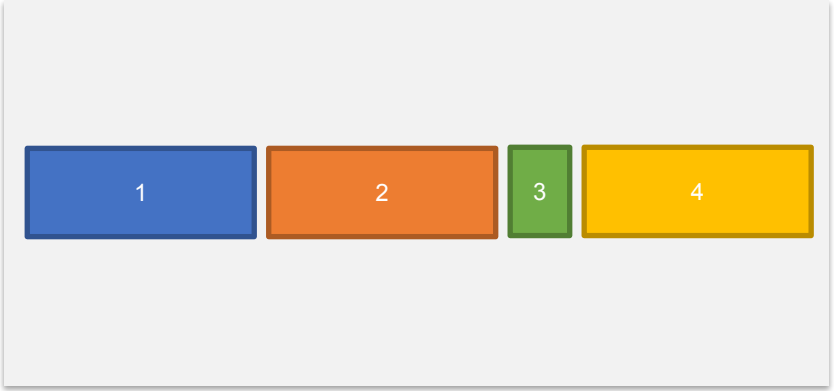
Flexbox Child Properties: flex-shrink

The **flex-shrink** property specifies the extent to which an item will shrink in relation to the other elements inside the flexbox container. It accepts a unitless value for proportion. For instance, in the following example, box3 will occupy three times **less** space than that occupied by the rest of the elements.

```
<div class="container">
   <div id="box1">1</div>
   <div id="box2">2</div>
   <div id="box3">3</div>
   <div id="box4">4</div>
</div>
```

```
.container {
   display: flex;
   width: 350px;
}

.container div { width: 100px; }
#box3 { flex-shrink: 3; }
```



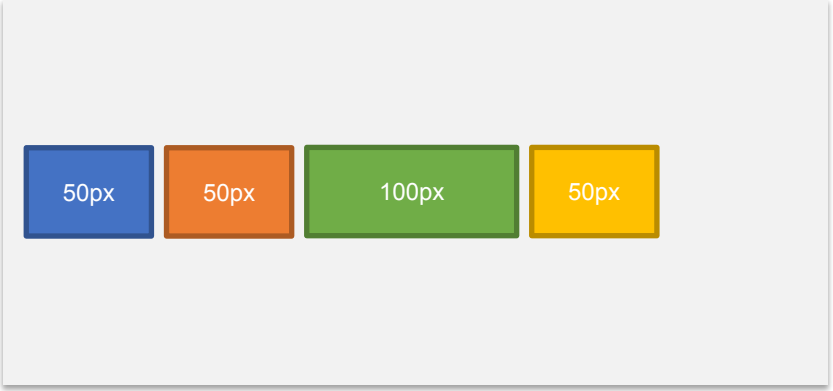
Flexbox Child Properties: flex-basis

The **flex-basis** property specifies the initial length of a flex item. It can also be set to *auto* or a number followed by *%*.

```
<div class="container">
   <div id="box1">1</div>
   <div id="box2">2</div>
   <div id="box3">3</div>
   <div id="box4">4</div>
</div>
```

```
.container {
   display: flex;
   width: 350px;
}

.container div { flex-basis: 50px; }
#box3 { flex-basis: 100px; }
```



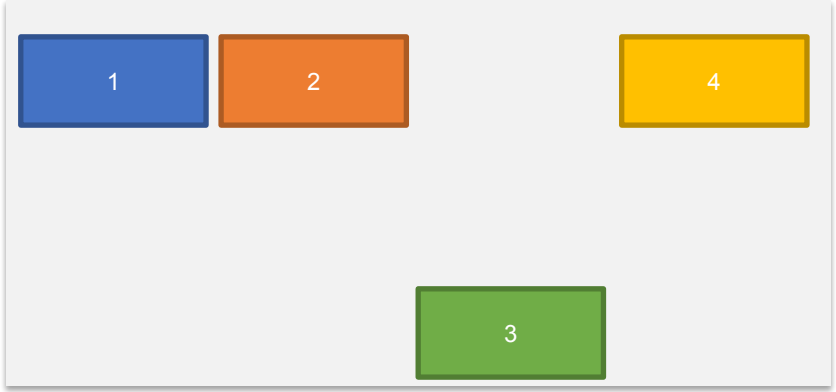
Flexbox Child Properties: align-self

The **align-self** property specifies the way in which an element should align itself in the flex container.

```
<div class="container">
   <div id="box1">1</div>
   <div id="box2">2</div>
   <div id="box3">3</div>
   <div id="box4">4</div>
</div>
```

```
html, body, .container {
   height: 100%;
   margin: 0;
}
.container {
   display: flex;
   width: 350px;
   align-items: flex-start;
}

#box3 { align-self: flex-end; }
```



Flexbox Child Properties: flex

The **flex** is a shorthand property which specifies ***flex-grow, flex-shrink, and flex-basis*** of the container.

```
<div class="container">
   <div id="box1">1</div>
   <div id="box2">2</div>
   <div id="box3">3</div>
   <div id="box4">4</div>
</div>
```

```
html, body, .container {
   height: 100%;
   margin: 0;
}
.container {
   display: flex;
}
#box3 { flex: 3 1 100px; }
```

Poll 4 (15 Sec)

For which of the following is ***flex*** a shorthand property?
(Note: More than one option may be correct.)

1. flex-grow
2. flex-shrink
3. flex-basis
4. flex-wrap

Poll 4 (Answer)

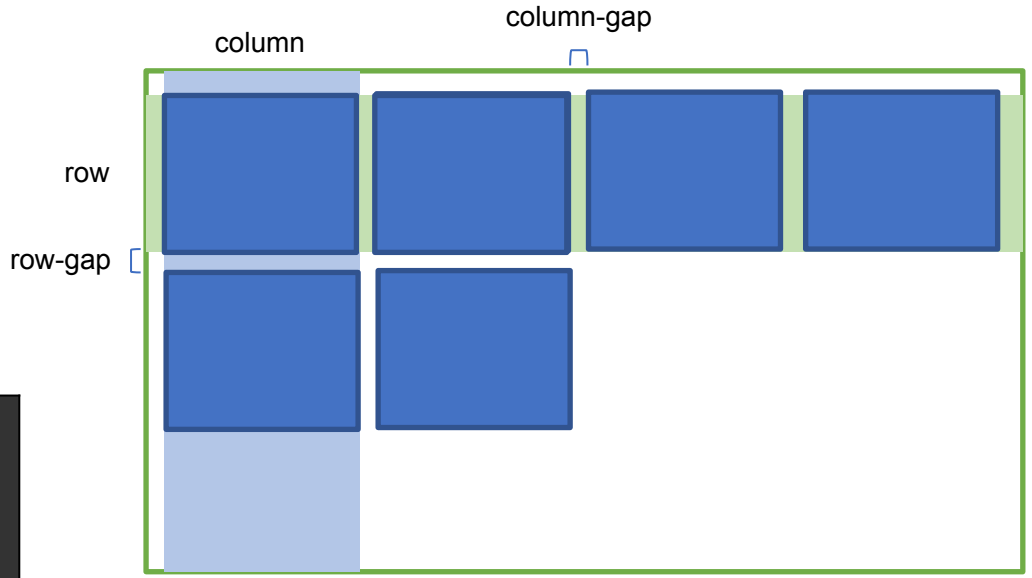
For which of the following is ***flex*** a shorthand property?
(Note: More than one option may be correct.)

1. **flex-grow**
2. **flex-shrink**
3. **flex-basis**
4. flex-wrap

CSS Grid

The CSS Grid Layout makes the layout into a grid format with rows and columns, which makes it easier to place elements rather than using float and positioning.

- Columns are the vertical lines of the grid items.
- Rows are the horizontal lines of the grid items.
- Grid gaps are the gaps between the columns and the rows.




```
.container {
   display: grid;
   grid-column-gap: 50px;
   grid-row-gap: 50px;
   /* grid-gap: 50px; If both row and column
gap should be same */
}
```

Positioning an Item on a Grid

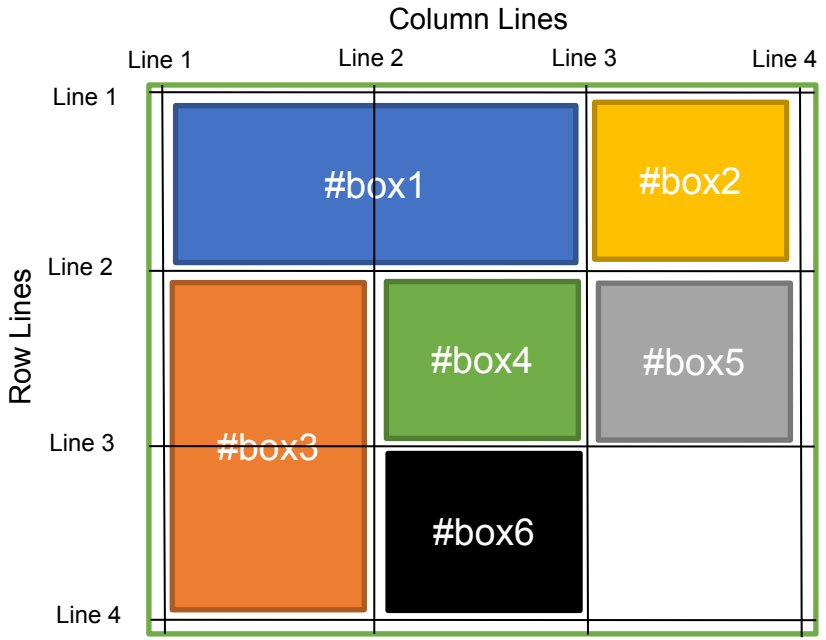
You can position a grid item on the grid using *grid-column-start*, *grid-column-end*, *grid-row-start* and *grid-row-end properties*.

- To position an element, imagine the lines for the column and rows as shown in the adjoining image.

- For instance, #box1 starts at line 1 and ends at line 3 and, hence, takes the width of of two grid columns. Similarly, #box3 takes the height of two grid rows.

```
#box1 {
   grid-column-start: 1;
   grid-column-end: 3;
}

#box3 {
   grid-row-start: 2;
   grid-row-end: 4;
}
```

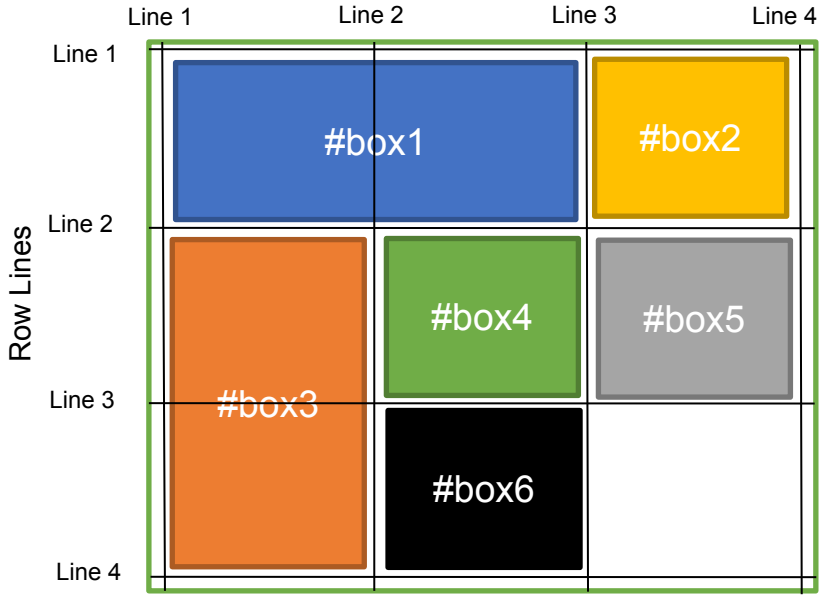


Grid Item

As you learnt earlier during positioning the grids, grid items can be positioned using *grid-column-start*, *grid-column-end*, *grid-row-start* and *grid-row-end properties*.

- A shorthand property for grid columns is *grid-column*, in which the start is separated from the end by /.
- A shorthand property for grid rows is *grid-row*, in which the start and the end are separated by /.

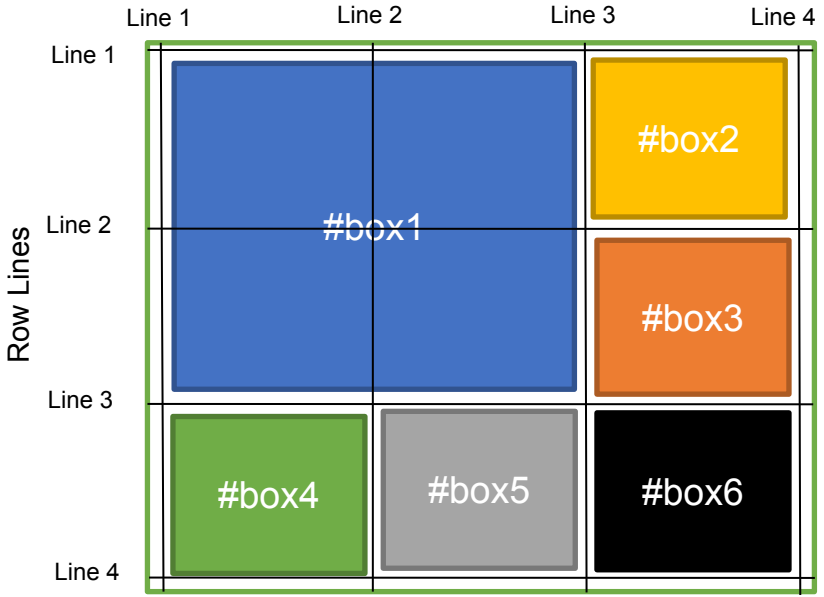
```
#box1 { grid-column: 1 / 3;}
#box2 { grid-column: 3 / 4;}
#box3 { grid-row: 2 / 4;}
#box4 { grid-row: 2 / 3;}
#box5 { grid-row: 2 / 3;}
#box6 { grid-row: 3 / 4;}
```



Grid Area

Using the **grid-area** property, you can assign all four properties together in the following order: separated by / – *grid-row-start / grid-column-start / grid-row-end / grid-column-end*.

```
#box1 {
   grid-area: 1 / 1 / 3 / 3;
}
```



Grid Container

To make an element behave as a grid-container, you need to input the following code: `.container { display: grid; }`

- The *grid-template-columns* property specifies the number of columns and the width of each one.
- For example, the code

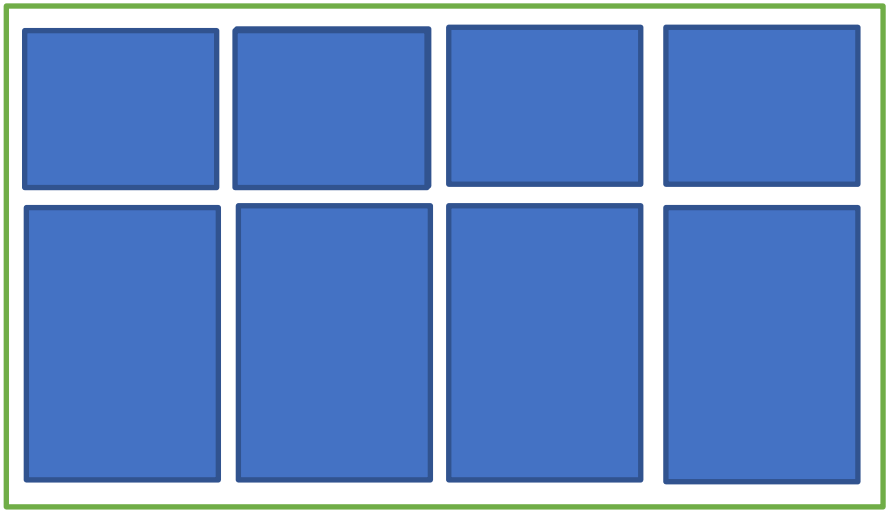
```
.container{ grid-template-columns: 50px 50px 50px 50px;}
```

of 50px each. You can replace 50px with auto to ensure that the columns have equal widths.

- Similarly, *grid-template-rows* specifies the number of rows and the height of each row.
- The code

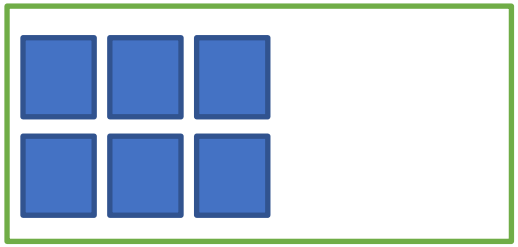
```
.container{grid-template-rows: 100px 250px;}
```

will create two rows in the grid, the first one with 100px height and the second one with 250px height.

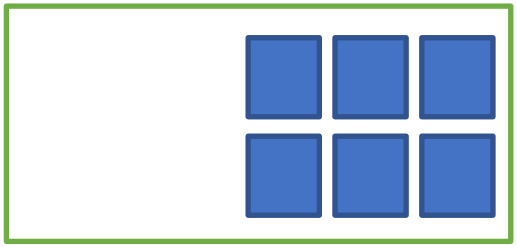


Grid Container: justify-content

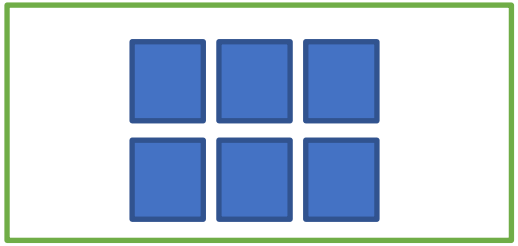
The **justify-content** property is used to align the grid items **horizontally** inside the container.
NOTE: The grid's total width needs to be lower than the container's width.



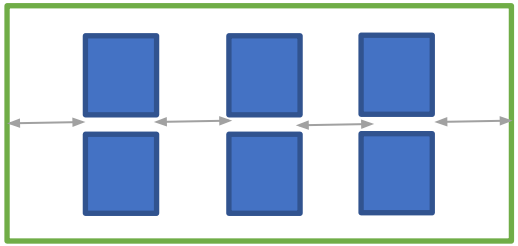
```
.container{ justify-content:
start;}
```



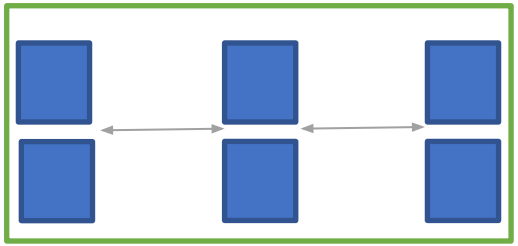
```
.container{ justify-content:
end;}
```



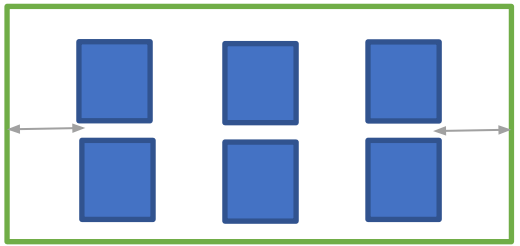
```
.container{ justify-content:
center;}
```



```
.container{ justify-content:
space-evenly;}
```



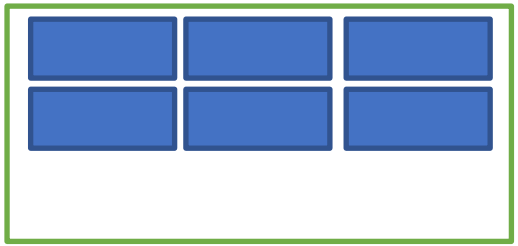
```
.container{ justify-content:
space-between;}
```



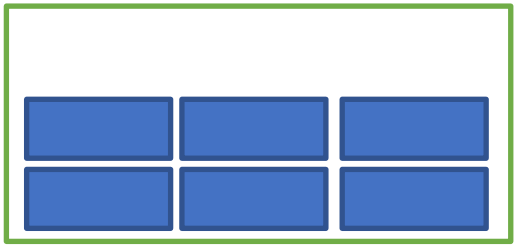
```
.container{ justify-content:
space-around;}
```

Grid Container: align-content

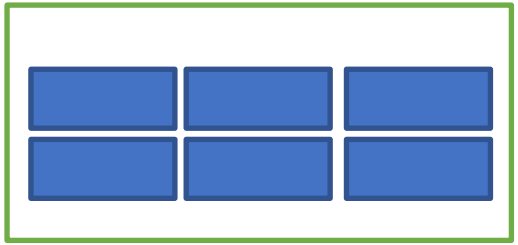
The **align-content** property is used to align the grid items **vertically** inside the container.
NOTE: The grid's total height needs to be lower than the container's height.



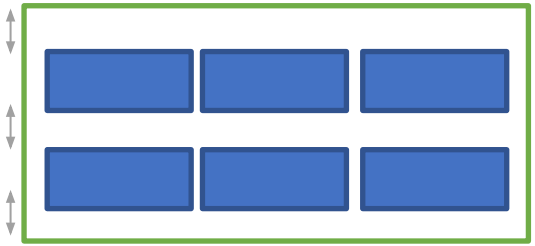
```
.container{ align-content:
start;}
```



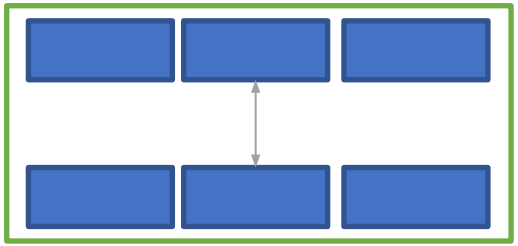
```
.container{ align-content: end;}
```



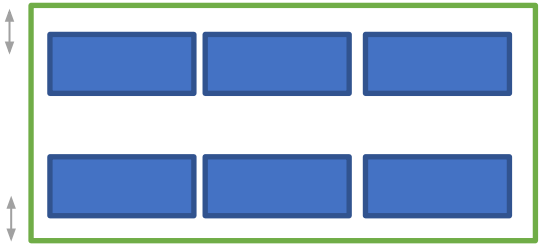
```
.container{ align-content:
center;}
```



```
.container{ align-content:
space-evenly;}
```



```
.container{ align-content:
space-between;}
```



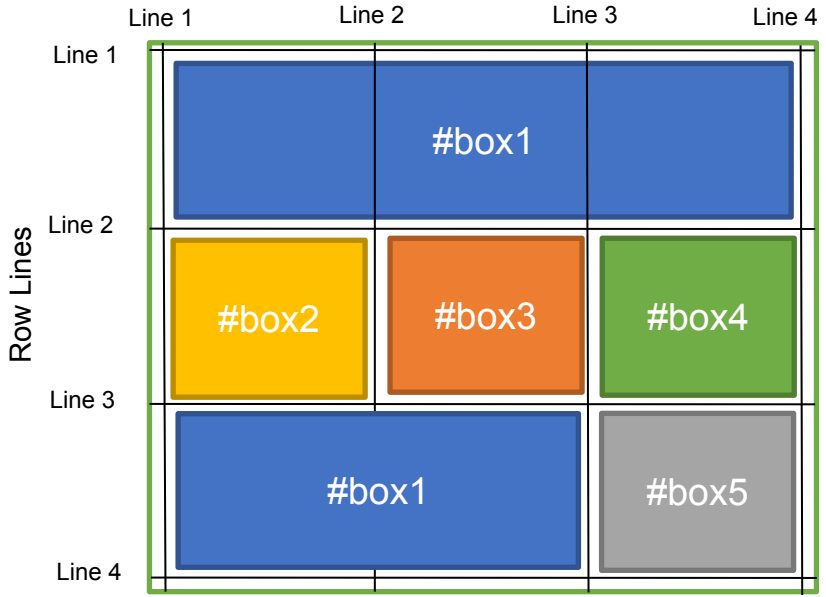
```
.container{ align-content:
space-around;}
```

Grid Area

Using the **grid-area** property, you can also assign a name for an area and use that area using **grid-template-areas** property. If the same area name is repeated consecutively, then it will become one merged area. You can use '.' to leave a grid empty for any other element.

```
#box1 {
   grid-area: "header";
}

.container{
   display: grid;
   grid-template-areas: "header header
header" "... " "header header .";
}
```



Poll 5 (15 Sec)

What is the output of the CSS command/tag in the adjoining image

1. The grid container has three columns with a width of 100px.
2. The grid container has three rows with a width of 100px.
3. Both 1 and 2
4. An error is thrown

```
.css-grid-container {
   grid-template-columns: 100px 100px
100px;
}
```

Poll 5 (Answer)

What is the output of the CSS command/tag in the adjoining image

1. **The grid container has three columns with a width of 100px.**
2. The grid container has three rows with a width of 100px.
3. Both 1 and 2
4. An error is thrown

```
.css-grid-container {
   grid-template-columns: 100px 100px
100px;
}
```

Project Work

(We will add the flexbox to our project.)

You can refer to the solution here.

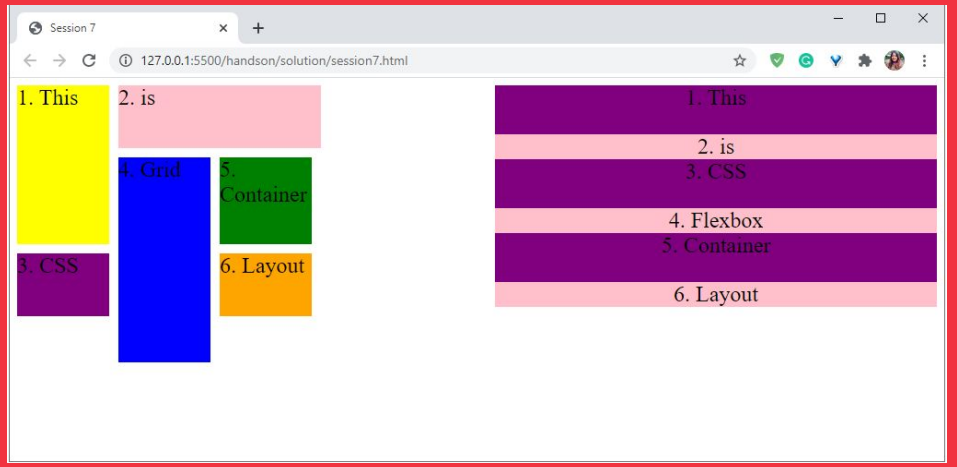
Hands-On Exercise (3 mins)

upGrad

Write a CSS code for the given HTML code such that the output is as shown in the adjoining screenshot.

1. The `<div class="grid-container">` is a grid container. Write the CSS for the grid container such that it looks similar to that in the adjoining screenshot.
2. The `<div class="flex-container">` is a flex container. Write the CSS for the flexbox such that it looks similar to that in the adjoining screenshot.

    a. Every even-numbered child of the flexbox has the background color pink.
    b. Every odd-numbered child of the flexbox has the background color purple.



The stub code is provided here.
The solution is provided here.

Key Takeaways from this class...

- A flexbox (flexible box) is used to adjust the width/height of elements so that they fit into containers. You can align items as columns or rows and even arrange the elements. 'You can grow or shrink the height/width of an individual element toadjust it according to the container.
- A CSS grid specifies a grid of row and columns to place elements.

upGrad

Doubt Clearance (5 mins)

The following tasks are to be completed after today's session:

MCQs
Coding Questions
Project - Checkpoint 7

In the next class, we will discuss...

- CSS3
- CSS Animations

upGrad

#RahoAmbitious

Thank you!