

Module 9 (Flex)

Module 9.1 (Learning Tables and Flex)

Que - how to create tables in html ?

Solution :

Table

Name	Class	Roll	Grade
Name	Class	Roll	Grade
Name	Class	Roll	Grade

```
<table id="my-table">

  <th>Main Table</th>

  <tr class="my-row">
    <td class="my-col">Name</td>
    <td class="my-col">Class</td>
    <td class="my-col">Roll</td>
    <td class="my-col">Grade</td>
  </tr>

  <tr class="my-row">
    <td class="my-col">Name</td>
    <td class="my-col">Class</td>
    <td class="my-col">Roll</td>
    <td class="my-col">Grade</td>
  </tr>

  <tr class="my-row">
    <td class="my-col">Name</td>
    <td class="my-col">Class</td>
    <td class="my-col">Roll</td>
    <td class="my-col">Grade</td>
  </tr>

</table>
```

```
#my-table{
  border : 1px solid black;
  font-size: 20px;
  width: 100%;
}

.my-row{
  border: 3px solid red;
}

.my-col{
  width: 25%;
  border: 1px solid red;
  text-align: center;
}
```

Que- What is Flex or Flexbox system and why it is used ?

Flex Box is basically a box or a system in which whatever elements we add will become inline element (not block).

initially we had some ways to make newspaper like structure of our website which were:-

- `position: absolute` and `left, right, top, bottom` for positioning
- `float: left/right`
- tables with rows and cols

but there is an issue as whenever a new element gets added in the webpage, it becomes a difficult task to place it at the right position, so to remove this stress we now use a new system

FLEX

Imp Points about About `display: Flex`

1. Flex is set on the display property.
2. lets say we have 3 elements and we need to place them in same line, then what we can do is simply place all of them inside a container (div).
3. Now set containers `display: flex` and suddenly all 3 elements will be inside the container and in same line.
4. There is also a `gap: 10px (value)` property which can be used to give space between the elements
5. Whenever we set the container's display to flex, what happens is all elements inside that container will become `inline` elements (not block element).
6. (VIP) the width of each element inside this flex box container will depend on their contents inside them (more context, more width) (but yes we can change the width)

- 6.2 The height of the container will by default depend on the size of content inside them, for example if we set height of each paragraph tag equal to 200px inside the container, then the container will also increase (but we can change this height anytime).

Learning Flex-Box

<p> Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore tempore repellendus dicta non facilis alias distinctio quam enim autem cum? </p>	<p> Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestias tenetur nobis quos porro unde quaerat? Suscipit vitae sint aperiam, quo alias tempora asperiores repellat perspiciatis illo iure maxime necessitatibus, deleniti cupiditate ipsam architecto sit labore, eum pariatur quam rerum esse ex! Cum modi consequatur fuga qui sed atque eveniet eum. </p>	<p> Lorem ipsum dolor sit, amet consectetur adipisicing elit. Iste, dignissimos? ipsum dolor sit amet consectetur adipisicing elit. Illum nostrum distinctio molestias assumenda odio ad, ipsa exercitationem quisquam labore ab alias atque neque aspernatur iure facere debitis! Fuga voluptatem ullam similique quae velit eveniet voluptate magnam alias, eius quia ex doloremque, in facilis expedita. Saepe </p>
---	--	--

breakdown:-

Learning Flex-Box

<p> Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore tempore repellendus dicta non facilis alias distinctio quam enim autem cum? </p>	<p> Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestias tenetur nobis quos porro unde quaerat? Suscipit vitae sint aperiam, quo alias tempora asperiores repellat perspiciatis illo iure maxime necessitatibus, deleniti cupiditate ipsam architecto sit labore, eum pariatur quam rerum esse ex! Cum modi consequatur fuga qui sed atque eveniet eum. </p>	<p> Lorem ipsum dolor sit, amet consectetur adipisicing elit. Iste, dignissimos? ipsum dolor sit amet consectetur adipisicing elit. Illum nostrum distinctio molestias assumenda odio ad, ipsa exercitationem quisquam labore ab alias atque neque aspernatur iure facere debitis! Fuga voluptatem ullam similique quae velit eveniet voluptate magnam alias, eius quia ex doloremque, in facilis expedita. Saepe </p>
---	--	--

```

<style>
    .container-div{
        display: flex;
        gap: 10px;
    }
    .paras{
        margin:0;
        background-color: aquamarine;
    }
</style>
</head>

<body>
    <h1>Learning Flex-Box</h1>

    <div class="container-div">

        <p class="paras">Lorem ipsum dolor sit amet consectetur adipisicing elit.
        Inventore tempore repellendus dicta non facilis alias distinctio quam enim autem cum?</p>
        <p class="paras">Lorem ipsum dolor sit amet consectetur adipisicing elit.
        Molestias tenetur nobis quos porro unde quaerat? Suscipit vitae sint aperiam, quo alias
        tempora asperiores repellat perspiciatis illo iure maxime necessitatibus, deleniti
        cupiditate ipsam architecto sit labore, eum pariatur quam rerum esse ex! Cum modi
        consequatur fuga qui sed atque eveniet eum.</p>
        <p class="paras"> Lorem ipsum dolor sit, amet consectetur adipisicing elit. Iste,
        dignissimos? ipsum dolor sit amet consectetur adipisicing elit. Illum nostrum distinctio
        molestias assumenda odio ad, ipsa exercitationem quisquam labore ab alias atque neque
        aspernatur iure facere debitis! Fuga voluptatem ullam similique quae velit eveniet
        voluptate magnam alias, eius quia ex doloremque, in facilis expedita. Saepe</p>
    </div>
  
```

```
<p class="paras">Lorem, ipsum dolor sit amet consectetur adipisicing elit.  
Cumque culpa fugiat, aut dolores, corporis provident at eum nostrum, explicabo similique  
officia rerum facere!</p>  
  
</div>
```

lets set width of each `p` to 25% (for 4 paras each)

Learning Flex-Box

Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore tempore repellendus dicta non facilis alias distinctio quam enim autem cum?

Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestias tenetur nobis quos porro unde quaerat? Suscipit vitae sint aperiam, quo alias tempora asperiores repellat perspiciatis illo iure maxime necessitatibus, deleniti cupiditate ipsam architecto sit labore, eum pariatur quam rerum esse ex! Cum modi consequatur fuga qui sed atque eveniet eum.

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Iste, dignissimos? ipsum dolor sit amet consectetur adipisicing elit. Illum nostrum distinctio molestias assumenda odio ad, ipsa exercitationem quisquam labore ab alias atque neque aspernatur iure facere debitis! Fuga voluptatem ullam similique quae velit eveniet voluptate magnam alias, eius quia ex doloremque, in facilis expedita. Saepe

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Cumque culpa fugiat, aut dolores, corporis provident at eum nostrum, explicabo similique officia rerum facere!

- 7. By default this flexbox container will be a block element, but suppose our containers contents are already wrapped in 50% of the width, so we want to use the remaining empty space of the container for other new elements by setting `display: inline-flex` (by doing this now the container's width by default will be equal to the content provided inside it), any other new element can also adjust in the remaining space.
- So basically inline-flex is used when we want our container and all inner child elements to have width = content present inside them

example1:-
container's

```
display: flex
```

note: each one of them inside the container are div

Red

Orange

Yellow

Green

Blue

Indigo

Purple



container's display: inline-flex



example2:-

container's display: flex:-

Learning Flex-Box

90%

Reset

Lorem ipsum dolor sit amet consectetur adipiscing elit. Invenire tempore repellendus dicta non facilis alias distinctio quam enim autem cum?	Lorem ipsum dolor sit amet consectetur adipiscing elit. Molestias tenetur nobis quos porro unde quaerat? Suscipit vitae sint aperiam, quo alias tempora asperiores repellat perspiciatis illo iure maxime necessitatibus aleniti cupiditate ipsam architecto	Lorem ipsum dolor sit amet consectetur adipiscing elit. Iste, dignissimos? ipsum dolor sit amet consectetur adipiscing elit. Illum nostrum distinctio molestias assumenda odio ad, ipsa exercitationen quisquam labore ab neque aspernatur iure facere dehitis!	Lorem, ipsum dolor sit amet consectetur adipiscing elit. Cumque culpa fugiat, aut dolores, corporis provident at eum nostrum, explicabo similique officia rerum facere!
--	--	---	---

container's display: inline-flex :-

Learning Flex-Box

Lorem ipsum dolor sit amet consectetur adipisicing elit. Inventore tempore repellendus dicta non facilis alias distinctio quam enim autem cum?	Lorem ipsum dolor sit amet consectetur adipisicing elit. Molestias tenetur nobis quos porro unde quaerat? Suscipit vitae sint aperiam, quo alias tempora asperiores repellat perspiciatis illo iure maxime necessitatibus alias atque deleniti cupiditate ipsam architecto sit labore, eum pariatur quam rerum esse ex! Cum modi consequatur fuga qui sed	Lorem ipsum dolor sit amet consectetur adipisicing elit. Iste, dignissimos? ipsum dolor sit amet consectetur adipisicing elit. Illum nostrum distinctio molestias assumenda odio ad, ipsa exercitationen	Lorem, ipsum dolor sit amet consectetur adipisicing elit. Cumque culpa fugiat, aut dolores, corporis provident at eum nostrum, explicabo similique officia rerum facere!
--	---	--	--

Code:-

```
<style>

  .container-div{
    border: 1px solid black;
    display: inline-flex;
    gap: 10px;
  }

  .paras{
    width: 80px;
    margin:0;
    background-color: aquamarine;
  }

</style>
```

</head>

<body>

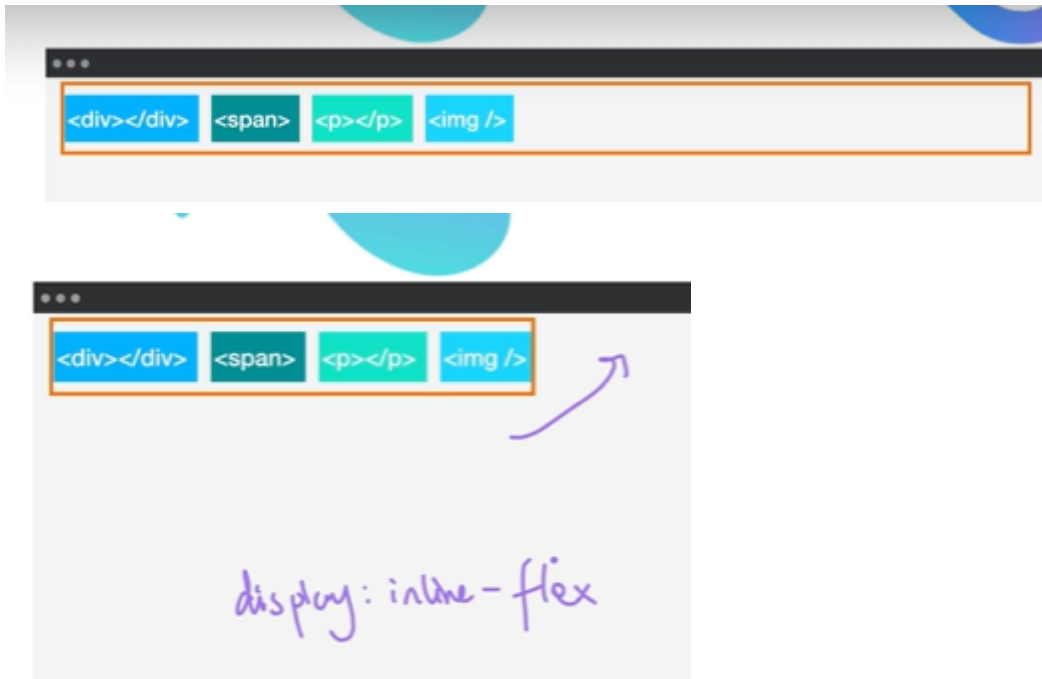
<h1>Learning Flex-Box</h1>

<div class="container-div">

```
<p class="paras">Lorem ipsum dolor sit amet consectetur
<p class="paras">Lorem ipsum dolor sit amet consectetur
<p class="paras"> Lorem ipsum dolor sit, amet consectet
<p class="paras">Lorem, ipsum dolor sit amet consectet
```

</div>

more examples:-

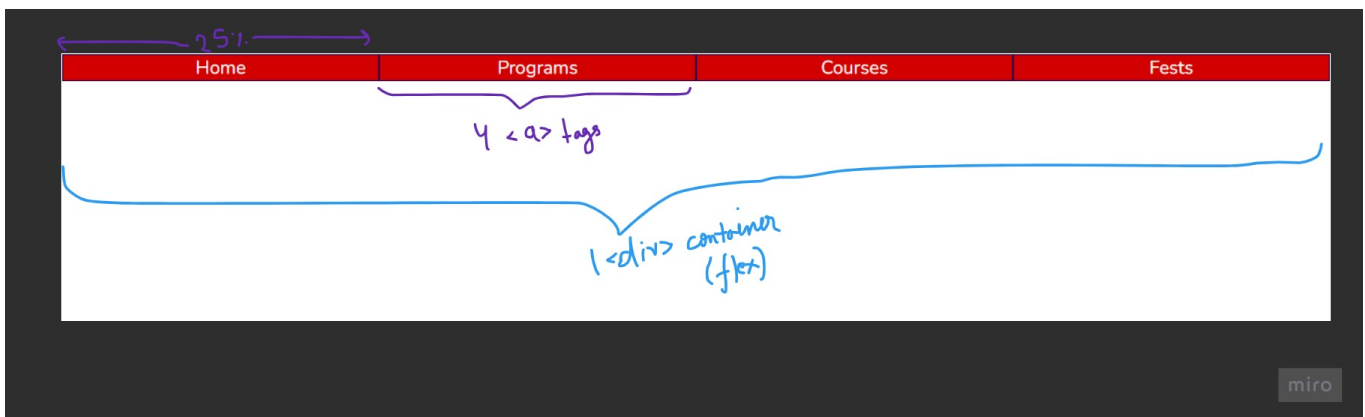


note: 🕒 now using this flex box we can very easily create nav bars, we will not have to use all those float, positioning and display:inline/block, etc. properties

Que - Create a navbar using only `display:flex` (do not use float, position, and other display values).



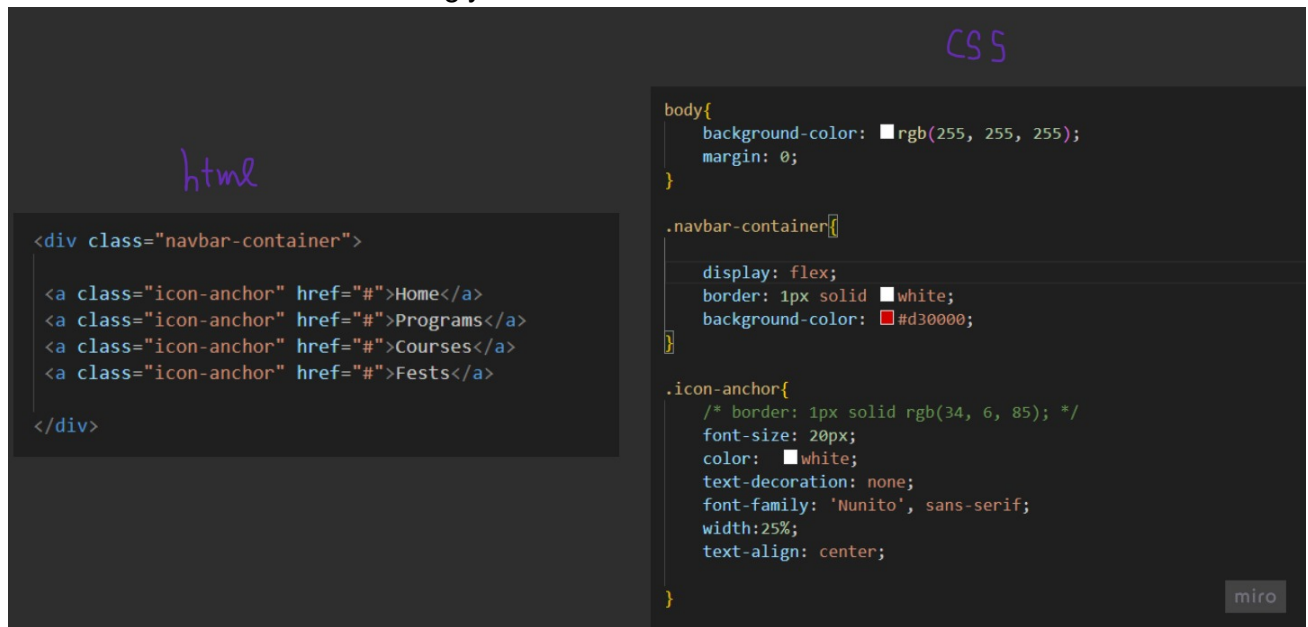
Breakdown :-



code:-

- By creating a div container, and inside that we have 4 anchor tags

- set display of container to flex
- after setting it to flex by default container will be block element.
- give all anchors width of (25%) each, now center their text using `text-align: center`
- give background-color of container, now increase the font size of the 3 anchors and the height of the flexbox will set itself accordingly.

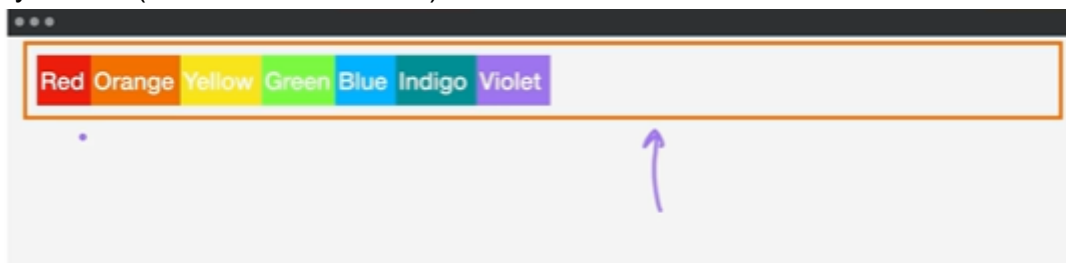


Module 9.2 (Flex-direction, Flex-Basis, main axis, crossaxis)

Que - What is flex-direction ?

- it is a property of 'flex-box', which specifies that in which direction the new elements inside the flexbox container will be added.

by default (flex-direction is = row)





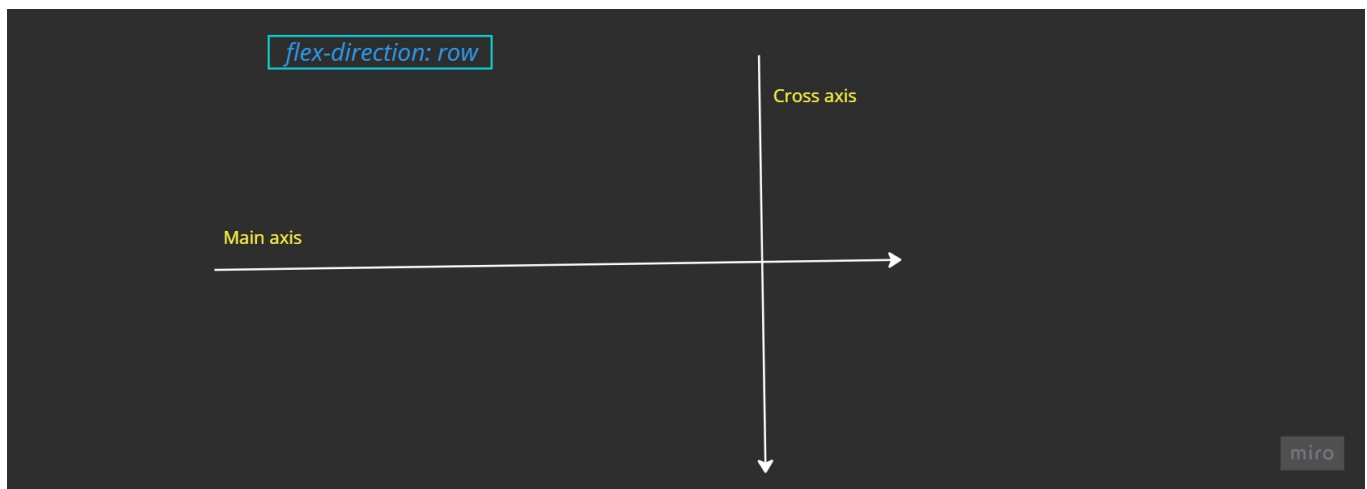
when we set `flex-direction: column`



- It is set on the parent (.container) element
- it's values: `flex-direction: row` and `flex-direction: col`.
- By default flex-direction is always set to row.
- There are 2 axis in flex box 1) main-axis 2) cross-axis.

When we set `flex-direction: row` all elements will be added in same row (untill its full)
i.e `main-axis` will move from left to right and `cross axis` will move from top to bottom.

When we set `flex-direction: col` all elements will be added in same col (untill its full)
i.e `main-axis` will move from top to bottom and `cross axis` will move from left to right.



But Why are these main-axis & cross axis important ?

Because There is another property `flex-basis: size` which we set on the containers child elements (or inner elements or items)

When we set containers (parent's) `flex-direction: row` (i.e main axis from left to right)
now when we set child's `flex-basis: 100px` then the width of all inner elements will increase to 100px

(width because main axis is now left to right)

```
.container {  
  border: 5px solid gold;  
  display: flex;  
  flex-direction: row;  
}  
  
.container > *{  
  flex-basis: 100px;  
}
```

When we set containers (parent's) `flex-direction: col` (i.e main axis from top to bottom) now when we set child's `flex-basis: 100px` then the height of all inner elements will increase to 100px (width because main axis is now top to bottom).

```
.container {  
  border: 5px solid gold;  
  display: flex;  
  flex-direction: column;  
}  
  
.container > *{  
  flex-basis: 100px;  
}
```

Que - Also What if we use display: inline-flex here?

Ans - then the element will take only that much width which is required for the content inside them



Que - Explain all of the 'flex-directions' with example.

- Red list is set to row
- Yellow list is set to row-reverse
- Blue list is set to column
- Green list is set to column-reverse



Chitkara University (Homepage) - Clone

i created a clone of CU Website homepage (not exact but yes many changes)



BTECH

Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptates, repellat. Lorem ipsum dolor sit amet consectetur adipisicing elit. Nesciunt, nihil! Perspiciatis similique, quis inventore officia magnam soluta blanditiis recusandae dolores suscipit, veniam dolorem quam! Culpa molestiae id maiores accusantium laudantium.

BBA

Lorem ipsum dolor, sit amet consectetur adipisicing elit. Animi blanditiis explicabo eos libero minus officia deleniti assumenda laboriosam impedit vero sed vel cumque officiis quisquam facere sunt atque dolore eius dolorem soluta ipsum maxime, temporibus possimus. Adipisci, iure. Quos, incidunt?

MBA

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Corrupti, sapiente labore vel inventore architecto asperiores molestias iste incidunt aperiam, optio voluptatum voluptatem ut totam iure suscipit autem aspernatur. Accusantium nisi eveniet quo magnam sequi optio nulla odio nobis itaque est.



© 2023 CU Clone by Yashasvi

<!-- H T M L Code -->

```
<html lang="en">
```

```
<head>
```

```
<meta charset="UTF-8">
```

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
<title>CU Clone</title>
```

```
<link rel="icon" href="./assets/pictures/chitkara-logo.png">
```

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

```
<link rel="preconnect" href="https://fonts.googleapis.com">
```

```
<!-- fonts from google -->
```

```

<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Varela&display=swap"
rel="stylesheet">
<link rel="preconnect" href="https://fonts.googleapis.com">
<link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
<link href="https://fonts.googleapis.com/css2?family=Open+Sans:wght@300;400&family=Varela&display=swap" rel="stylesheet">
</head>

<body>

<div class="navbar-container">
  <a href="#" class="navbar-anchor">Admissions</a>
  <a href="#" class="navbar-anchor">Courses</a>
  <a href="#" class="navbar-anchor">Fees</a>
  <a href="#" class="navbar-anchor">Reviews</a>
  <a href="#" class="navbar-anchor">Placements</a>
  <a href="#" class="navbar-anchor" id="contact-us-anchor">Contact Us</a>
</div>

<!-- chitkara logo -->
<div class="main-logo-container">
  
</div>

<!-- chitkara campus img -->
<div class="campus-pic-container">
  
</div>

<!-- courses cards -->
<div class="courses-cards-outer-container">

  <div class="courses-cards-inner-container">

    <div class="course-card">
      <h2>BTECH</h2>
      <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptates,
repellat. Lorem ipsum dolor sit amet consectetur adipisicing elit. Nesciunt, nihil!
Perspiciatis similique, quis inventore officia magnam soluta blanditiis recusandae
dolores suscipit, veniam dolorem quam Culpa molestiae id maiores accusantium laudantium.
</p>
    </div>

    <div class="course-card">
      <h2 class="course-card-heading">BBA</h2>
      <p>Lorem ipsum dolor, sit amet consectetur adipisicing elit. Animi
blanditiis explicabo eos libero minus officia deleniti assumenda laboriosam impedit vero

```



```
sed vel cumque officiis quisquam facere sunt atque dolore eius dolorem soluta ipsum  
maxime, temporibus possimus. Adipisci, iure. Quos, incidunt?</p>
```

```
</div>
```

```
<div class="course-card">
```

```
<h2>MBA</h2>
```

```
<p>Lorem ipsum dolor sit, amet consectetur adipisicing elit. Corrupti,  
sapiente labore vel inventore architecto asperiores molestias iste incidunt aperiam,  
optio voluptatum voluptatem ut totam iure suscipit autem aspernatur. Accusantium nisi  
eveniet quo magnam sequi optio nulla odio nobis itaque est.</p>
```

```
</div>
```

```
</div>
```

```
</div>
```

```
<!-- footer -->
```

```
<footer class="footer-container">
```

```
<iframe class="chitkara-gmap" src="https://www.google.com/maps/embed?  
pb=!1m18!1m12!1m3!1d184821.4673014666!2d76.60220702728469!3d30.599150057419063!2m3!1f0!2f  
0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x390fc32344a6e2d7%3A0x81b346dee91799ca!2sChitkar  
a%20University!5e0!3m2!1sen!2sin!4v1690727333097!5m2!1sen!2sin" width="600" height="450"  
style="border:0;" allowfullscreen="" loading="lazy" referrerpolicy="no-referrer-when-  
downgrade"></iframe>
```

```
<p class="footer-para">
```

```
© 2023 CU Clone by Yashasvi
```

```
</p>
```

```
</footer>
```

```
</body>
```

```
</html>
```

```
/* CSS CODE */
```

```
body{  
    margin:0;  
}
```

```
/* -----NavBar Styling----- */
```

```
.navbar-container{  
    /* border: 1px solid black; */  
    background-color: #d00000;  
    display: flex;  
    flex-direction: row;  
    /* gap: 10px; */  
}
```

```

.navbar-anchor{
  text-align: center;
  color: white;
  text-decoration: none;
  font-size: 18px;
  width: 16%;
  margin-top: 10px;
  margin-bottom: 10px;
  font-family: 'Varela', sans-serif;
  /* border: 1px solid pink; */
}

#contact-us-anchor{
  margin: 0;
  /* border: 1px solid white; */
  background-color: rgb(247, 247, 79);
  color: black;
  padding-top: 8px; /*just to center the anchor*/
}

.navbar-anchor:hover{
  color: rgb(0, 255, 234);
}

#contact-us-anchor:hover{
  background-color: rgb(43, 100, 93);
  color: rgb(241, 241, 241);
}

/* -----Chitkara center main logo----- */

.chitkara-logo-img{
  height: 200px;
  /* border: 1px solid black; */
  width: 70%;
  margin-left: 15%;
  margin-right: 15%;
  object-fit: contain;
}

/*----- chitkara campus img -----*/

.campus-pic-container{
}

.chitkara-campus-img{
  width: 90%;
}

```

```

    margin-left: 5%;
    margin-right: 5%;
    border-radius: 40px;
}

/* -----courses cards----- */
/* note i created 2 containers because, it is difficult to center 3 divs inside a single
container, but using 2 containers, what i did is i centered the inner container (width:80%
leftpush:10% rightpush:10%) and inside this inner container are my 3 divs (Cards), then i
applied flex-row on the inner container

*/

.courses-cards-outer-container{
    padding-top: 40px;
    padding-bottom: 40px;
    width: 100%;
    /* border: 1px solid black; */
    background-color: #f5f5f5;
    margin-top: 20px;
    margin-bottom: 20px;
    gap: 20px;
}

.courses-cards-inner-container{
    width: 80%;
    margin-left: 10%;
    margin-right: 10%; /*to center this inner container(which is inside outer
container)*/
    /* border: 3px solid rgb(211, 25, 25); */
    display: flex;
    flex-direction: row; /* so that we can fit the 3 cards equally inside the inner-
container*/
    gap: 20px;
}

.course-card{
    flex-basis: 33%;
    /* border: 1px solid black; */
    background-color: white;
    border-radius: 20px;
    padding: 10px;
    text-align: center;
    font-family: 'Open Sans', sans-serif;
}

/*----- footer maps----- */

.footer-container{

```

```
background-color: black;
height: 300px;
width: 100%;
}

.chitkara-gmap{
width: 60%;
margin-left: 20%;
margin-right: 20%;
margin-top: 40px;
height: 150px;
border-radius: 20px;
}

.footer-para{
text-align: center;
color: white;
font-family: 'Open Sans', sans-serif;
font-size: 13px;
}
```

Module 9.3 (Flex-Layout)

There are a lot of properties to set the different layouts for a flexbox items

```
- order : value;
- flex-wrap : nowrap | wrap | wrap-reverse;
- justify-content: flex-start | flex-end | center | space-between | space-around | space-between
/*used on main-axis */
- align-items : (works on nowrap)
- align-self:
/* used on cross-axis*/
- align-content : (works on wrap)
```

1. order: of flex

- this property is set on the child element always (whose order we want to change).

By default the order of items inside flexbox container is 0, so that is the reason that when we add new elements in the container, they add on from the end of the flex (if flex-direction is row, then flex-end will be right most), and element will keep on adding in this order . Once the line is full, then elements will add in new lines.

now what if we do not want this default behavior rather we want some item/items to be in different order then we can use this `order:` property to set order of that item

The image shows three panels illustrating the `order` property in CSS flexbox.

Panel 1: A flex container with three items: "1.Red" (red), "2.Blue" (blue), and "3.Green" (green). Below each item is a handwritten "0".

Panel 2: The same flex container, but the order is changed. The items are "3.Green" (green), "1.Red" (red), and "2.Blue" (blue). Below each item is a handwritten "-1". A code block shows the CSS for the green item:

```
.green-div{
  background-color: green;
  order: -1;
}
```

Panel 3: The same flex container, but the order is changed again. The items are "3.Green" (green), "2.Blue" (blue), and "1.Red" (red). Below each item is a handwritten "1", "2", and "3" respectively. A code block shows the CSS for all three items:

```
.red-div{
  background-color: red;
  order: 3;
}
.blue-div{
  background-color: blue;
  order: 2;
}
.green-div{
  background-color: green;
  order: 1;
}
```

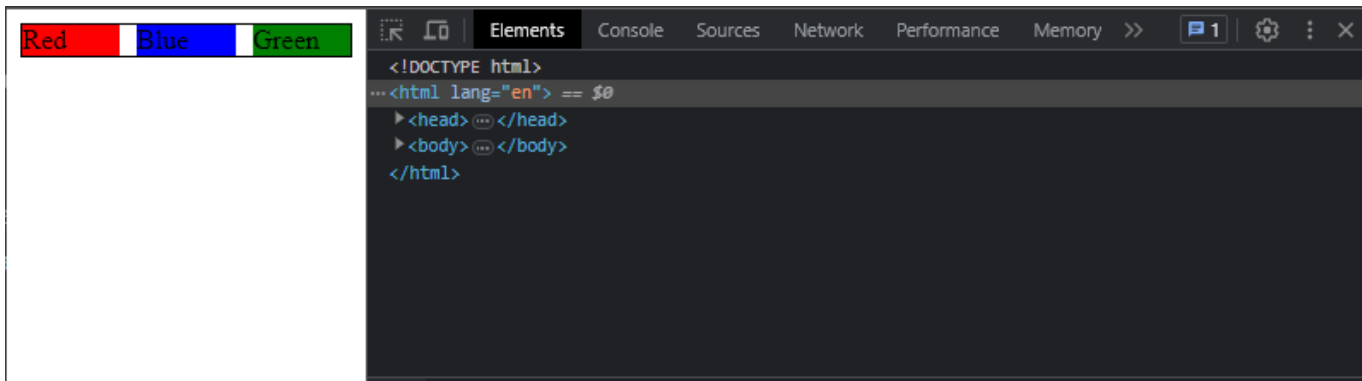
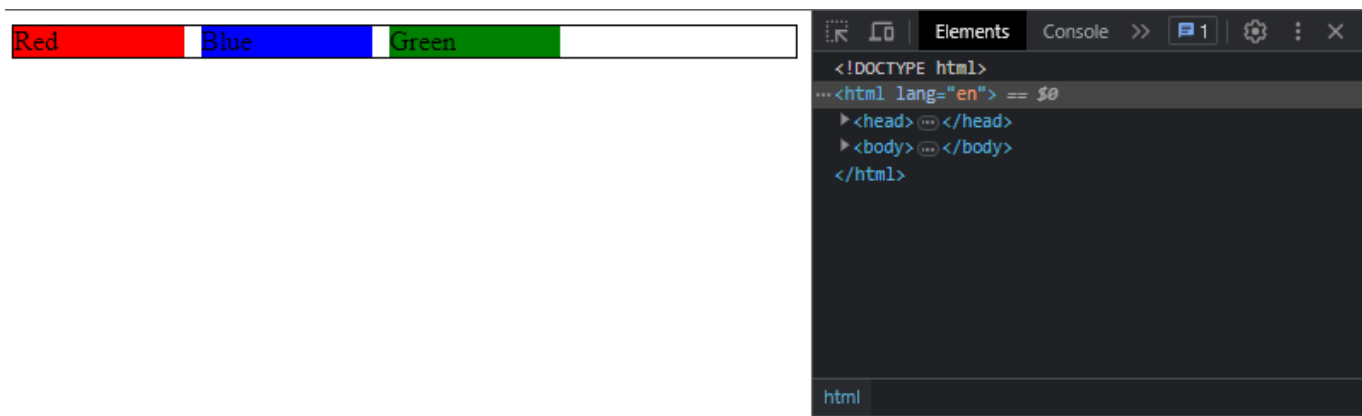
2. `flex-wrap: wrap | nowrap | wrap-reverse`

- It is basically used to make sure that whether the child's inside parent container should wrap or not when the screen size hits their width;
- this property is set on the parent container always.

flex-wrap: nowrap

by default every flex-box container has `flex-wrap: nowrap` set.

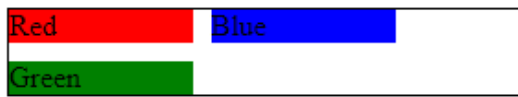
here when the screen size hits width of the child element in the corner, then all the child elements inside the flexbox container will start to shrink.



flex-wrap: wrap;

if we set `flex-wrap: wrap` then whenever the screen end hits the last child element's width ,then it will go on to the next line of the container, and if we continue to shorten the screen then each element will go on to the next line untill we stop decreasing.





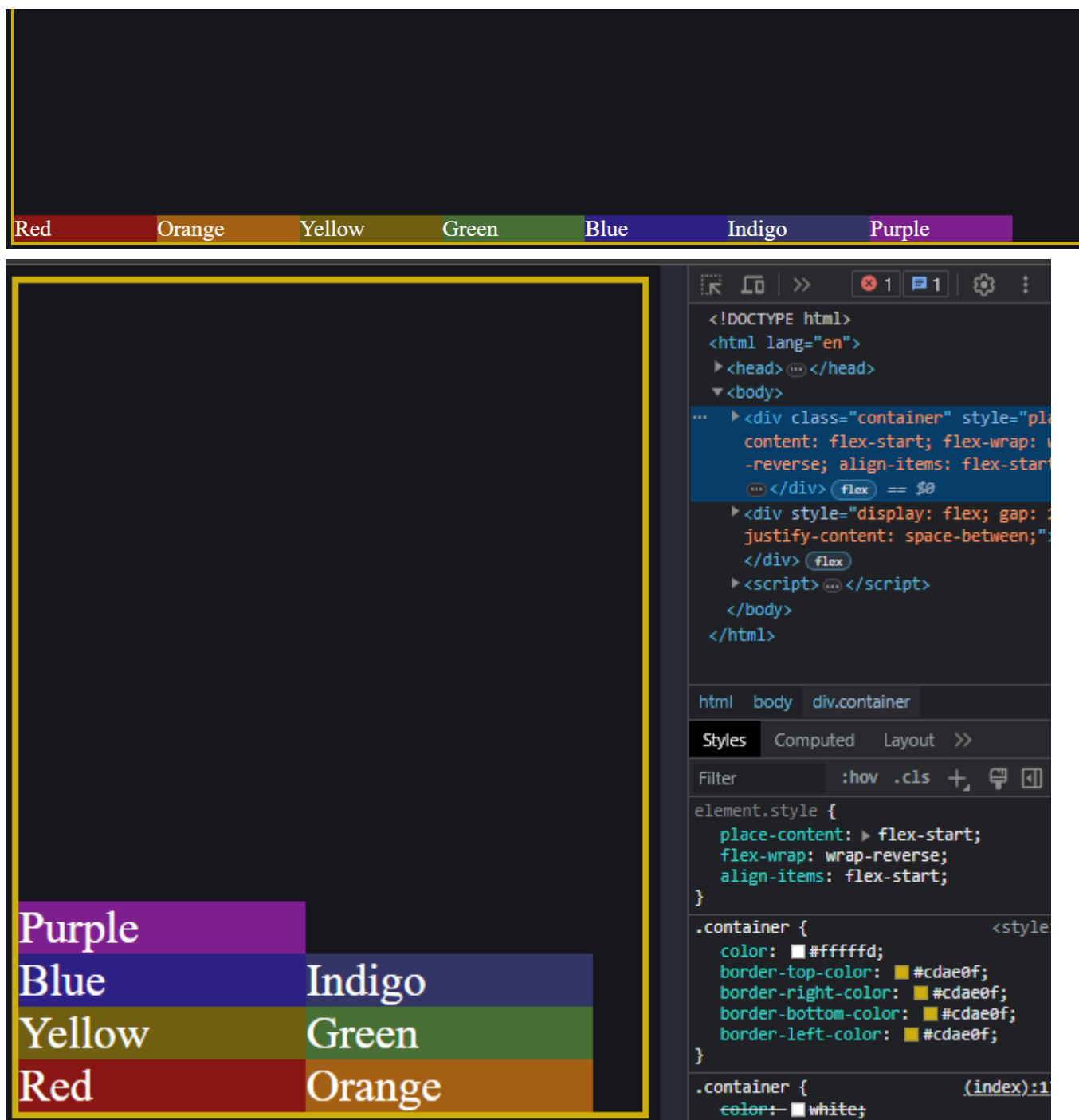
```
Elements Console Sources Network >>
<!DOCTYPE html>
...<html lang="en"> == $0
  ><head> ... </head>
  ><body> ... </body>
</html>
```



```
Elements Console Sources Network Performance Memory >>
<!DOCTYPE html>
...<html lang="en"> == $0
  ><head> ... </head>
  ><body> ... </body>
</html>
```

flex-wrap: wrap-reverse

this will do the same thing but in order bottom to top, i.e when the screen shrinks the item (child) will go a new upper line.



3. justify-content :

- This property is used to define the position of the items (child) in the center
- (I M P) works both on `wrap` and `nowrap`
- It is applied on the parent (container) element always
- It works always as per the main axis.
- (I M P) If the main axis is from left to right then the `flex-start` is left corner, `flex-end` is right corner. and if the main axis is from top to bottom, then `flex-start` is top and `flex-end` is bottom.

values -> flex start, flex-end, center, space-between, space-around, space-evenly.

center:

`flex-start`: (by default 'left' for `flex-direction: row`) and ('top' for `flex-direction: column`)

`flex-end`: vice versa of `flex-start`

`center` : it is one of the best ways and easiest to center items inside a flex-box container

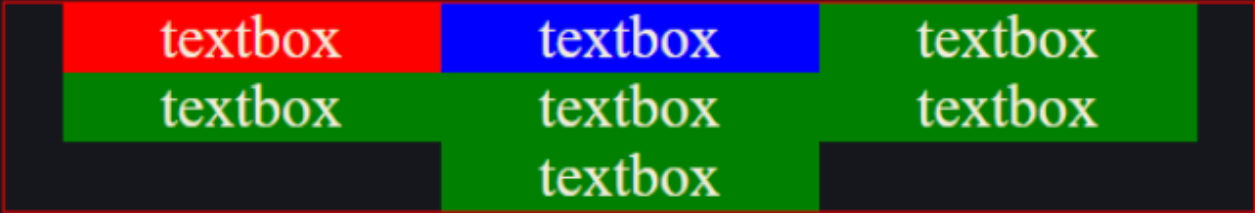
`space-between`: all items will have equal space only between them

`space-around`: all items will have its own margin (space) on their left and right corners, and also at the start and end

`space-evenly`: equal spaces between elements and start and end as well.

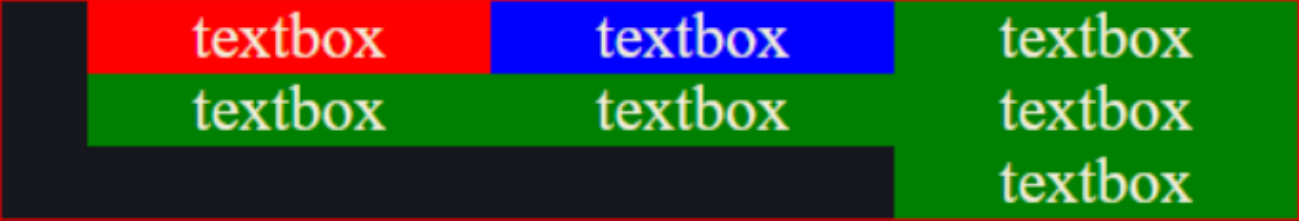


Que - What if items is not only in 1 line, then how will this 'justify-content' work?



The diagram shows a container with three flex items: a red 'textbox', a blue 'textbox', and a green 'textbox'. These items are wrapped into two lines. The first line contains the red and blue textboxes, and the second line contains the green textbox. The items are centered horizontally within the container.

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

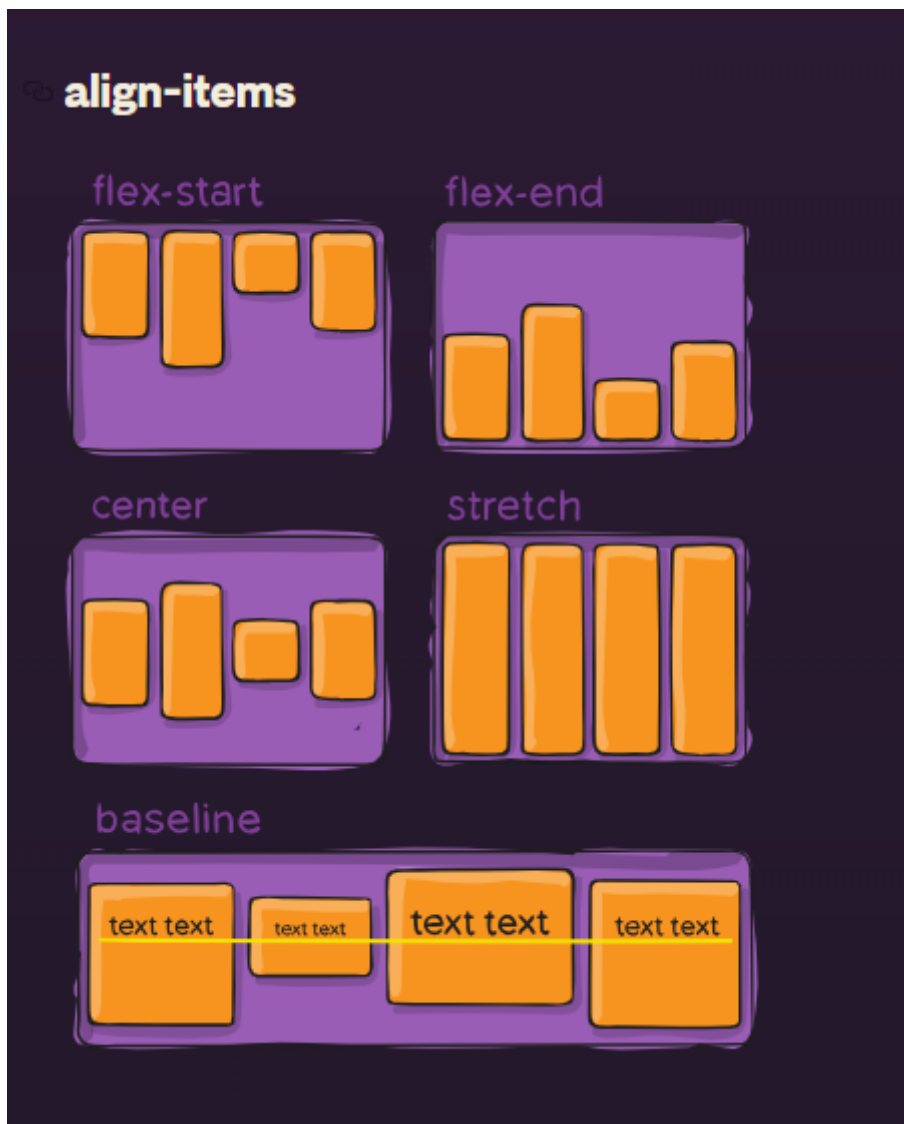


The diagram shows a container with three flex items: a red 'textbox', a blue 'textbox', and a green 'textbox'. These items are wrapped into two lines. The first line contains the red and blue textboxes, and the second line contains the green textbox. The items are aligned to the right side of the container.

```
.container{  
  flex-wrap: wrap;  
  justify-content: right;  
}
```

4. align-items :

- This defines the default behavior for how flex items are laid out along the `CROSS Axis`
- Think of it as the `justify-content` version for the cross-axis (perpendicular to the main-axis).
- works only in `flex-wrap: nowrap` . (of `flex-wrap` property) because if we wrap then items will go to next line.
- if `flex-direction: row` that means main axis from left to right and cross axis from top to bottom , so in that case `flex-start` is the top and `flex-end` is bottom end of the container.



What is `vh` and how it is used as a unit for height ?

`vh` means viewport height, viewport means height of screen so if we write `height: 70vh` then height of our element will set to 70% of the total screen's height.

5. align-content

- works in cross axis
- works on `flex-wrap: wrap`

- works even with multiple lines of items.

textbox

textbox

textbox

align-content: flex-start;

align-content: flex-end;

textbox

textbox

textbox

textbox

textbox

textbox

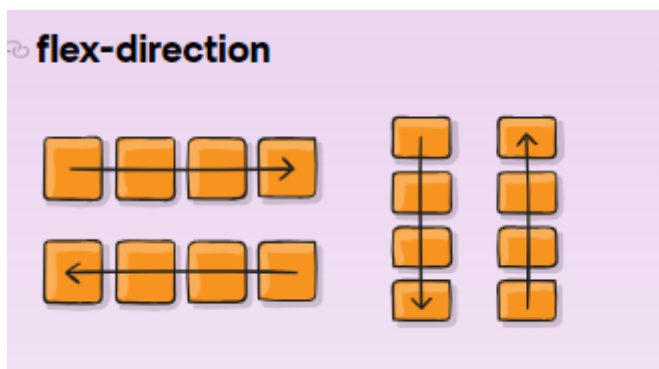
align-content: center;



Que - How to reverse the order of the items inside flexbox container ?

Ans - using `flex-direction: row-reverse;` or `flex-direction: column-reverse;`

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



Notice : that when you set the direction to a reversed row or column, start and end are also reversed.

Module 9.4 (Flex Sizing)

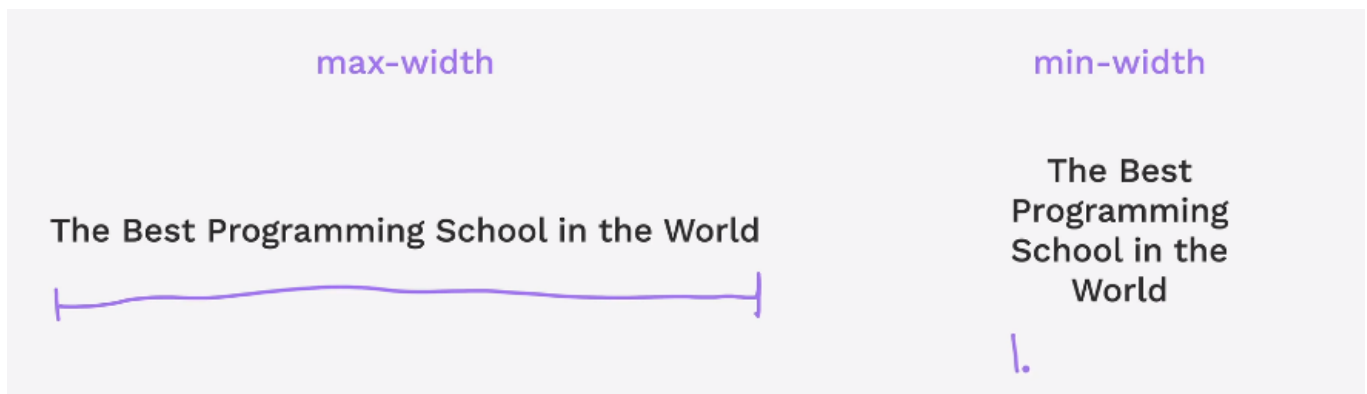
By default when we create lets say 4 div then they will be block elements, as soon as we put them into a flexbox container, then the div items will become inline element, now when we reduce the width size of this container, the div items will also start to shrink (as soon as the with of container hits the last element of the div items)

In most cases this is what we want, i.e we want out items to shrink, but sometimes we want a different type of sizing, so for that we can customize the sizing which will be based on the priority basis :-

```
content-width < width < flex-basis < min-width/max-width
```

1. content-width

when we compress the container, then the min width of div item will be equal to the width of longest word of that div items, and max width will be equal to the width of all content combined inside that div item



2. width

when we set width of items to lets say 100px, then all inner div will have width of 100px initially but if we shrink the container div to an extent that it hits the last item div then, the items will start on shrinking again.

3. Flex-basis

- applied on container

We already learned flex basis earlier that it works on the (main-axis) always.

flex basis has more priority then the normal `width` and `content-width`.

so if we set the `flex-direction=row` and `flex-basis` to 100px then by default the width will be 100px.

now if we shrink the view port width, and if it hits the last div, then it will start to shrink again and this shrinking will stop ones the width of each item div reaches the `content-width`.

4. min-width/max-width

- (IMP) these properties are applied on the child items (not the flexbox container)

note: if we apply these properties on the `container{}` div then it will not work for the items, it will work for container only, so use these properties on the child elements

`Max width` property is maximum possible width of div items, now

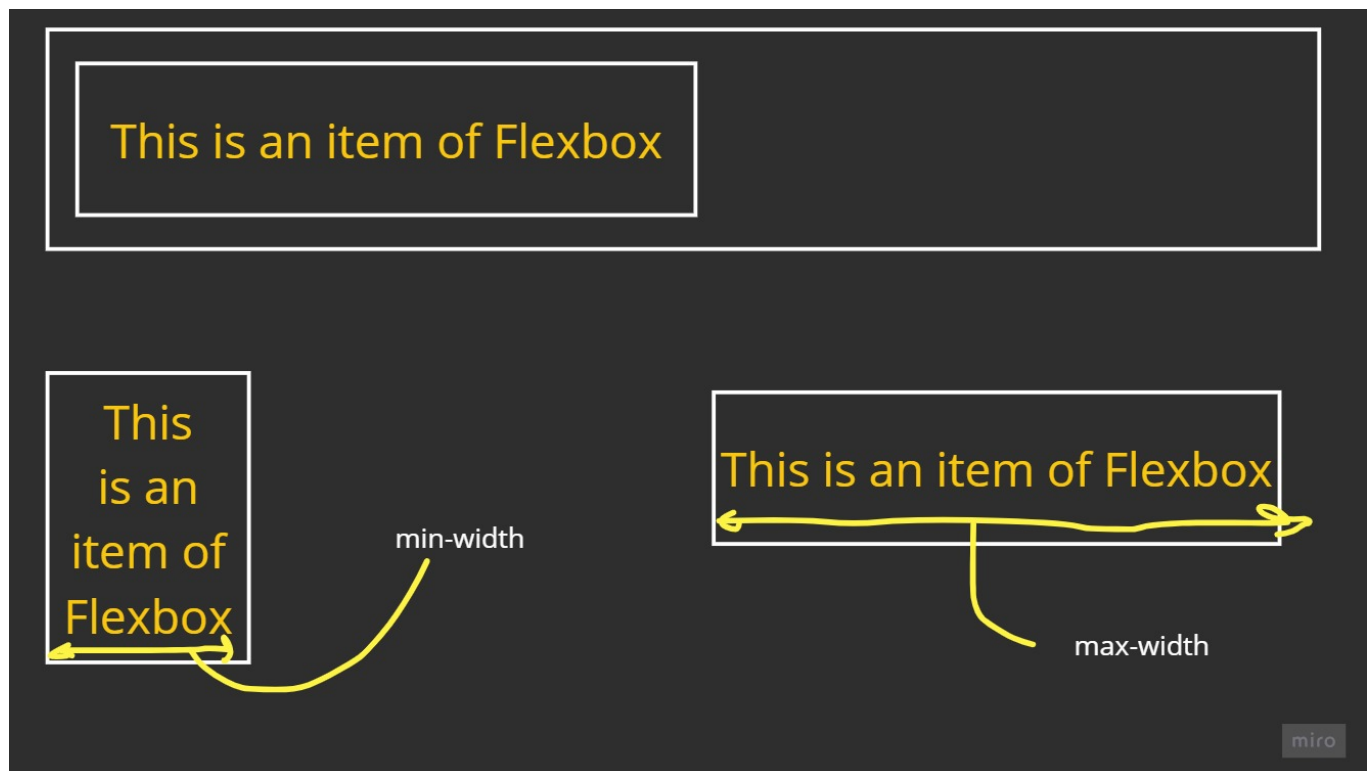
if `max-width: 100px` and `flex-basis: 50px` then div items will have width of 100px

if `max-width: 50px` and `flex-basis: 100px` then div items will have width of 100px

similarly if `min-width` is larger then the `flex-basis` then `min-width` will be respected

by default for a `<div> This is an item of Flexbox</div>`

- `min-width` will be equal to the width of longest word in the item i.e. `Flexbox` .
- `max-width` will be equal to the width of complete content inside that item

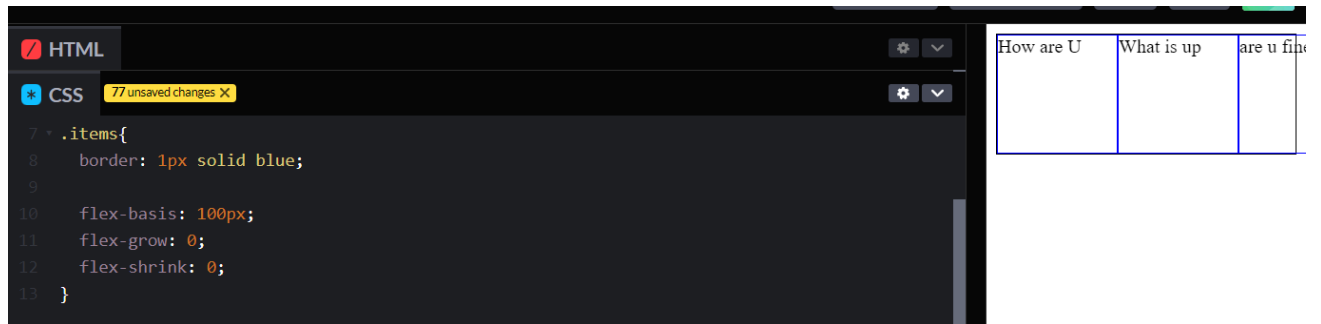
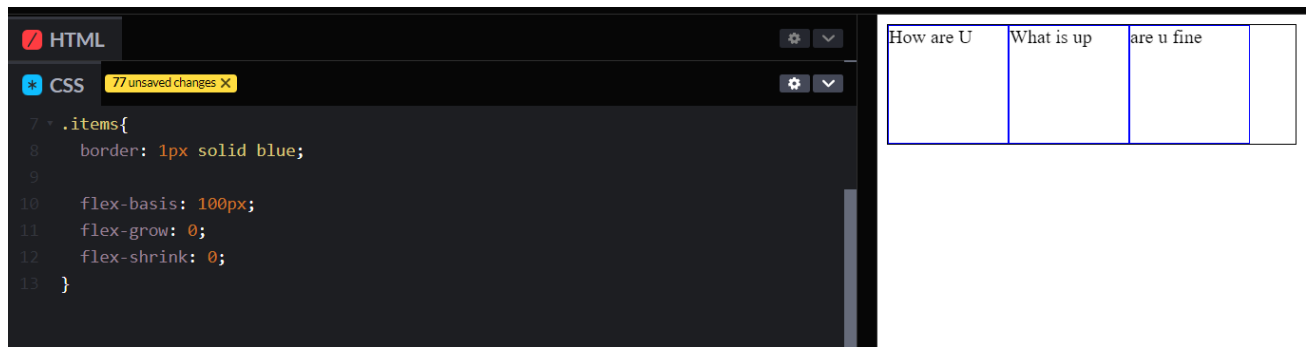


What is `flex-grow` & `flex-shrink` ?

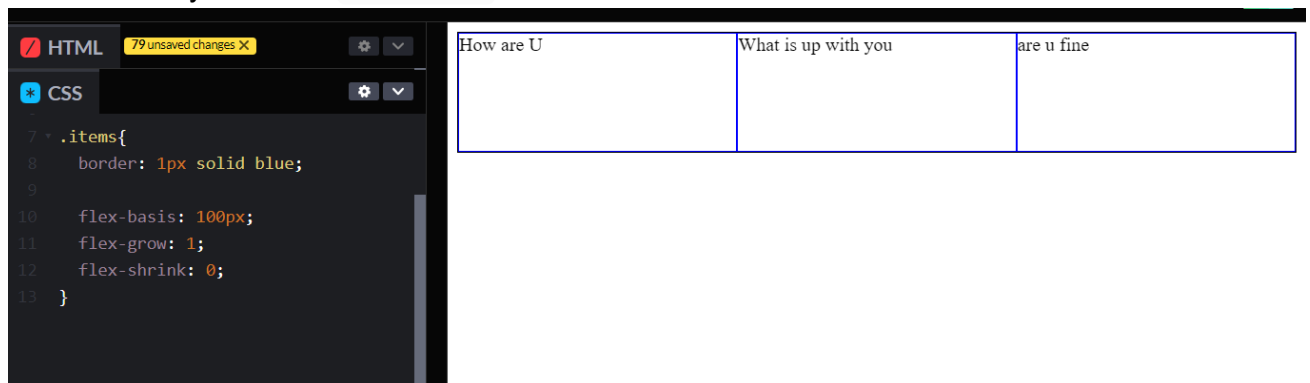
- applied on the `items`

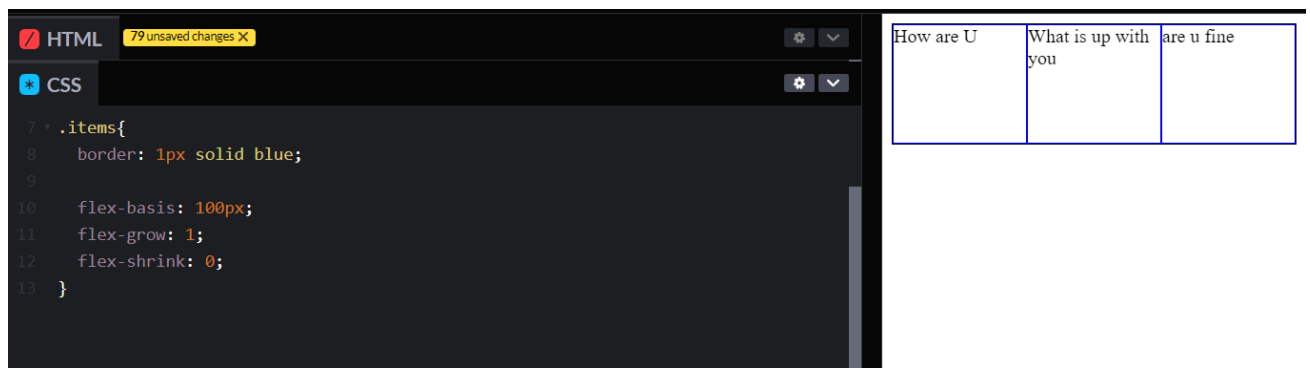
Flexbox mainly works on its `flex-grow` and `flex-shrink` property, let's understand it better:-

1. let's set `flex-grow: 0` and `flex-shrink: 0` then the width of each item will be equal to the `flex-basis` value always, and items will neither grow nor shrink, i.e even if we reduce the width of view port then items will not shrink.

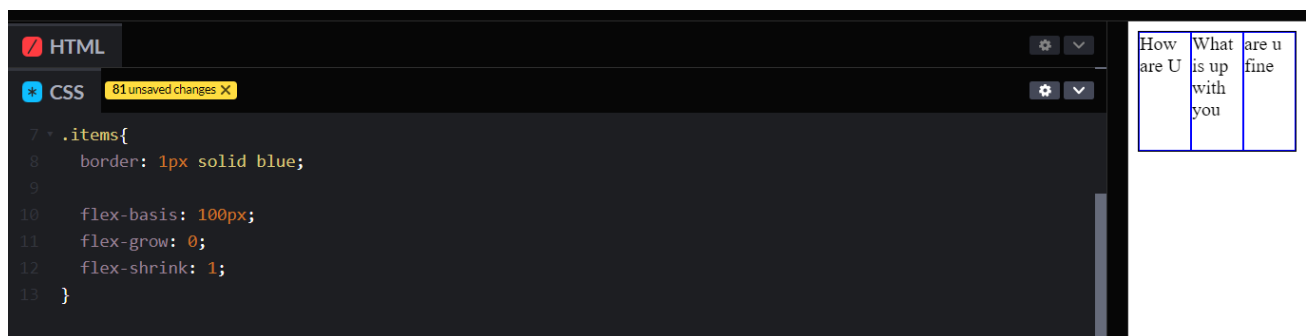
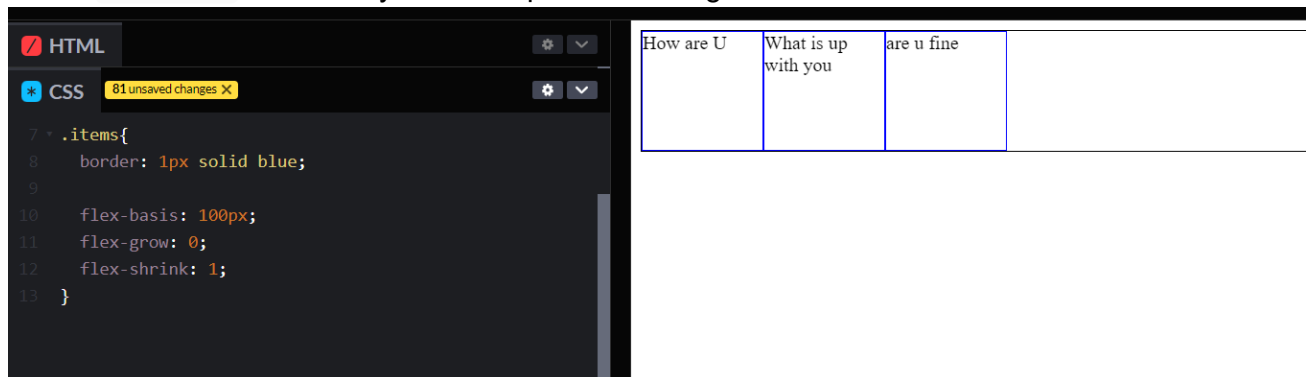


2. lets set `flex-grow: 1` and `flex-shrink: 0` then in this case the width of each item will first adjust according to the flex-basis, but because it is allowed to grow then it will grow and each item will have equal width with no space left out in the container div, but as we shrink the view port width, then it will only shrink till `flex-basis` set items



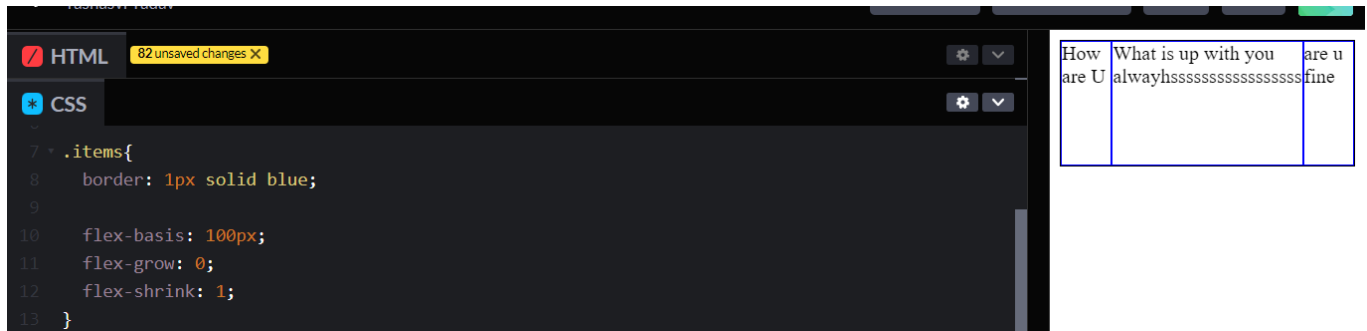
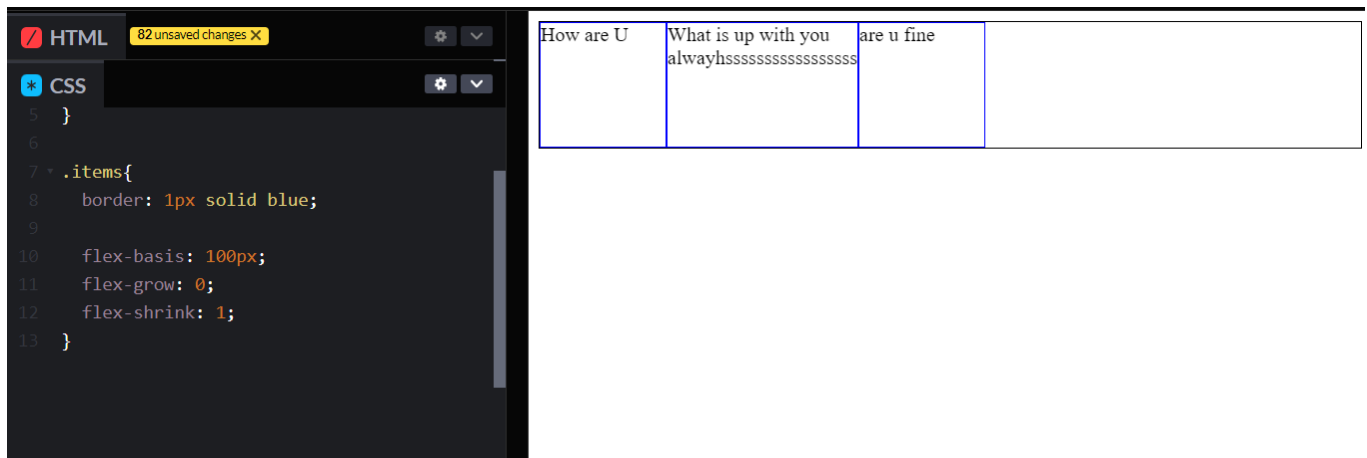


3. lets set `flex-grow: 0` and `flex-shrink: 1` then in that case again the width of each item will start with flex basis i.e 100px here, then since it is not allowed to grow so the width will stop at 100px but it is allowed to shrink that means if we reduce the viewport width then items will shrink till the `min-width` which is by default equal to the longest word's width inside each item

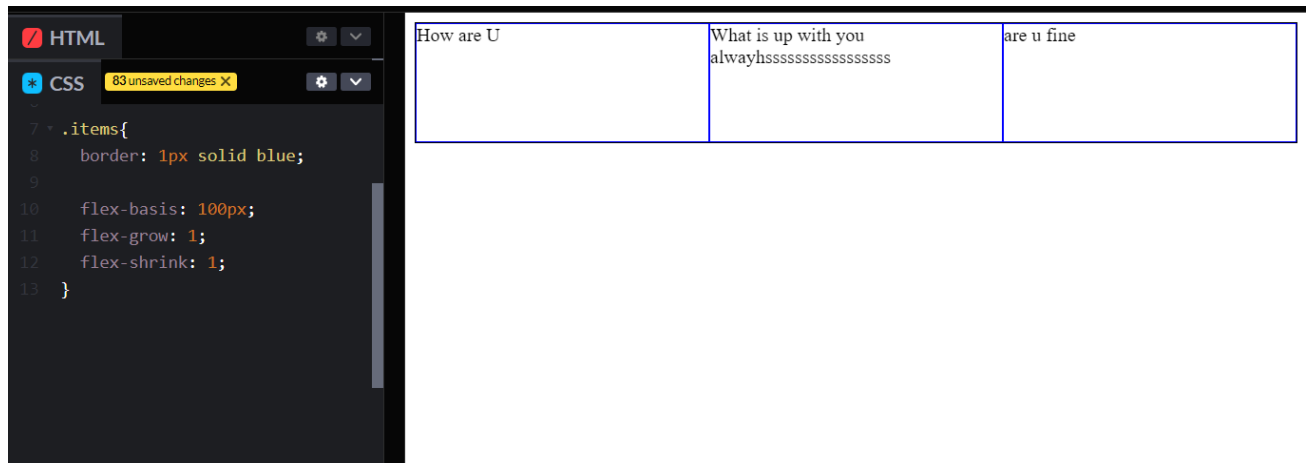


eg.2

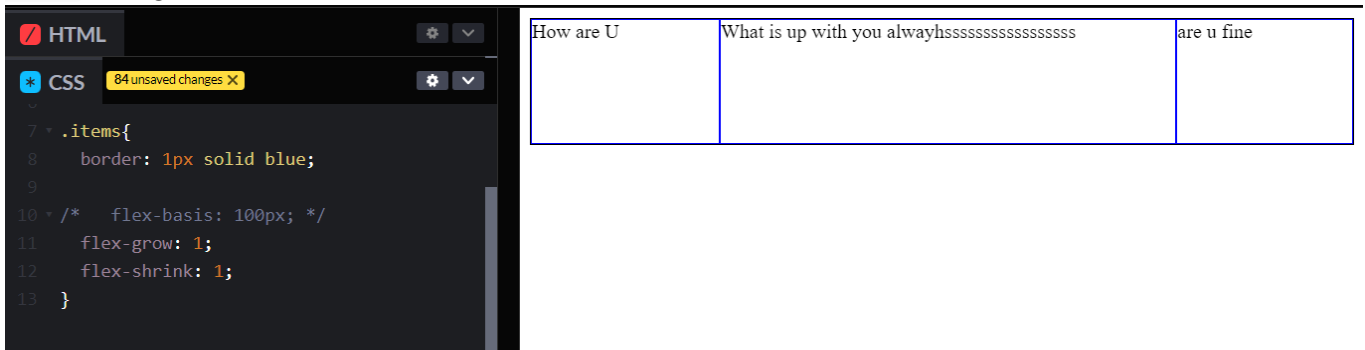
notice below that the width of each item will first be 100px (flex-basis) but as the min-width by default is equal to the width of longest word in each div item then the 2nd div will grow till its min-width (min-width is the width from which the div item stops to shrink)



4. Lets set `flex-grow: 1` and `flex-shrink: 1` then in that case first the width of each item will start with 100px (in our case) but for 2nd div it will be more (as discussed in previous example) and because it is allowed to grow, then each item will fill up the remaining space in the container equally, and also when we reduce the viewport width, then since it is allowed to shrink, then each div item will shrink till it reaches their min-width (not after that).



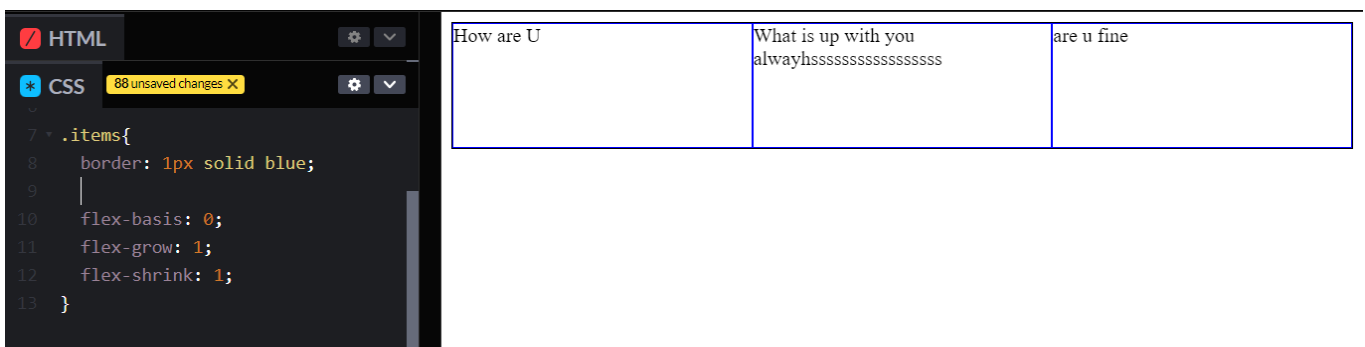
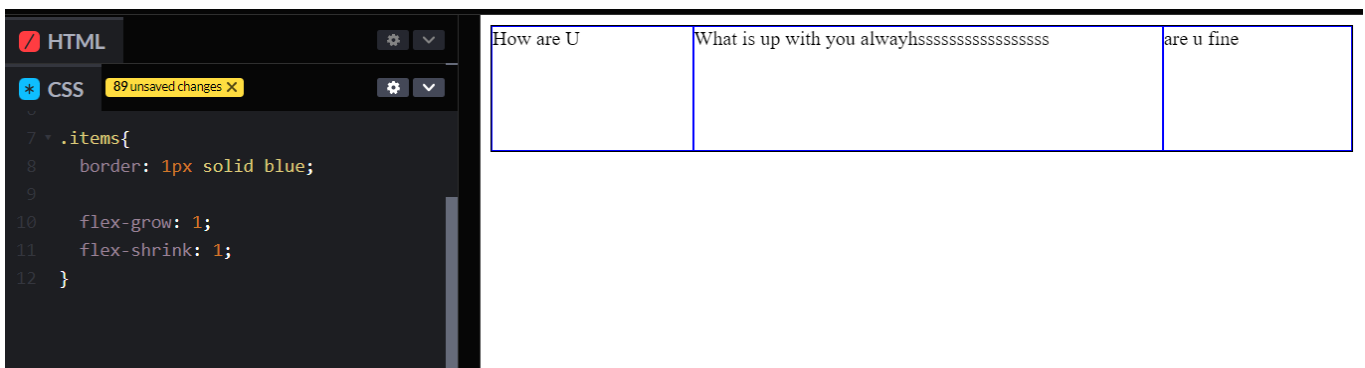
note: in case we have not set the flex-basis property, then by default the width will shrink till min-width and grow till max-width



what is flex-basis: auto ?

By default flex basis is always auto and what it does is that inside the flex-box the width of each item will be given according to the amount of content provided inside them.

(IMP) if we do not want that then, and want all our items to have equal width then set flex-basis: 0 ;



Que - What is short hand way to write these flex properties?

```
.items{
    flex: 1 1 0; /* grow:1 shrink:1 flex-basis:0 */
}
```

```
.items{
    flex: 1 0 100px; /* grow:1 shrink:0 flex-basis:100px */
}
```

```
.items{
    flex: 1; /*grow,shrink,flex-basis all with value 1*/
}
```

How to give width of each item, relative to the other items using `flex` ?

HTML

CSS 100+ unsaved changes

```

10 * .item1{
11     flex: 1;
12 }
13 * .item2{
14     flex: 2;
15 }
16 * .item3{
17     flex: 3;
18 }

```

How are U	What is up with you alwayhssssssssssssssssss	are u fine
-----------	---	------------

HTML

CSS 100+ unsaved changes

```

10 * .item1{
11     flex: 1;
12 }
13 * .item2{
14     flex: 5;
15 }
16 * .item3{
17     flex: 10;
18 }

```

How are U	What is up with you alwayhssssssssssssssssss	are u fine
--------------	---	------------

This will divide entire width into $1+5+10 \Rightarrow 16$ columns and allot 1st col to 1st div item, next 5 to 2nd div and remaining 10 columns space to last div item.

note: it is responsive.

HTML

CSS 100+ unsaved changes X

```
10 .item1{
11   flex: 1;
12 }
13 .item2{
14   flex: 5;
15 }
16 .item3{
17   flex: 10;
18 }
```

How	What is up with you	are u fine
are	alwayhssssssssssssssss	
U		

Exercise to practice flex-sizing : <https://appbrewery.github.io/flexbox-sizing-exercise>
