Cascading Style Sheet (CSS)

Internet Programming I: Chapter 3 – Part III

ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY

Department of Software Engineering

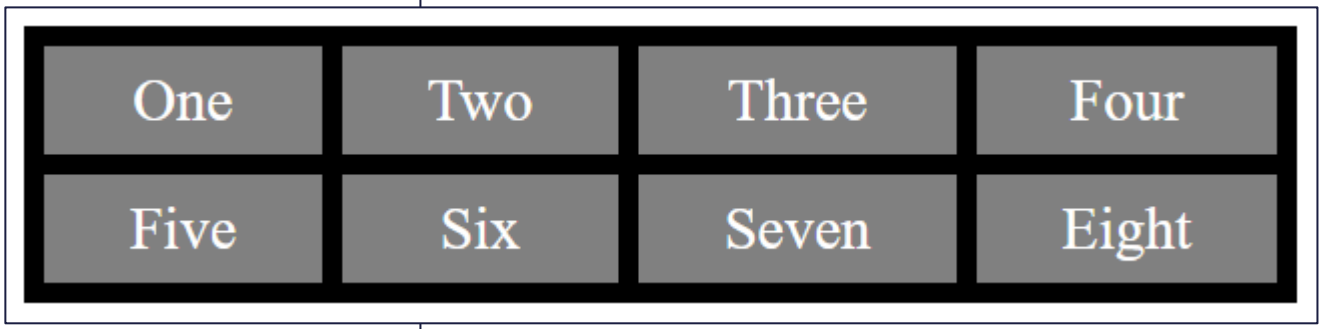
Main Source: https://www.javatpoint.com/

CSS Grid

- CSS Grid layout divides a page into major regions
- Grid property offers a grid-based layout system having rows and columns
- It makes the designing of web pages easy without positioning and floating
- Similar to the table, it enables a user to align the elements into rows and columns
  - But compare to tables, it is easy to design layout with the CSS grid
  - We can define columns and rows on the grid by using grid-template-rows and grid-template-columns properties
- Grid Container
  - We can define the grid container by setting the display property to grid or inline-grid on an element
  - Grid container contains grid items that are placed inside rows and columns

Grid Container example

```
.main {
        display: grid;
        grid: auto auto / auto auto auto auto;
        grid-gap: 10px;
        background-color: black;
        padding: 10px;
    }
    .num {
        background-color: grey;
        text-align: center;
        color: white;
        padding: 10px 10px;
        font-size: 30px;
    }
....
  <div class="main">
    <div class="num">One</div>
    <div class="num">Two</div>
    <div class="num">Three</div>
    <div class="num">Four</div>
    <div class="num">Five</div>
    <div class="num">Six</div>
    <div class="num">Seven</div>
    <div class="num">Eight</div>
  </div> .....
```



CSS Grid shorthand properties

- **grid-template-columns**: It is used to specify the size of the columns
- **grid-template-rows**: It is used to specify the row size
- **grid-template-areas**: It is used to specify the grid layout by using the named items
- **grid-auto-rows:** It is used to specify the automatic size of the rows
- **grid-auto-columns**: It is used to specify the automatic size of the columns
- **grid-auto-flow**: It is used to specify how to place auto-placed items and the automatic row size

CSS Grid Example using shorthand properties

```
…….
      .main {
         display: grid;
         grid-template-columns: 1fr 1fr 1fr;
         grid-template-rows: 100px 250px 200px;
         background-color: black;
         grid-gap: 10px;
         padding: 20px;
      }
      .num {
         background-color: lightgrey;
         text-align: center;
         padding: 20px 10px;
         font-size: 30px;
      }
   </style>
…..
   <div class="main">
      <div class="num">One</div>
      <div class="num">Two</div>
      <div class="num">Three</div>
      <div class="num">Four</div>
      <div class="num">Five</div>
      <div class="num">Six</div>
      <div class="num">Seven</div>
      <div class="num">Eight</div>
   </div> …….
```

CSS Grid cont'd

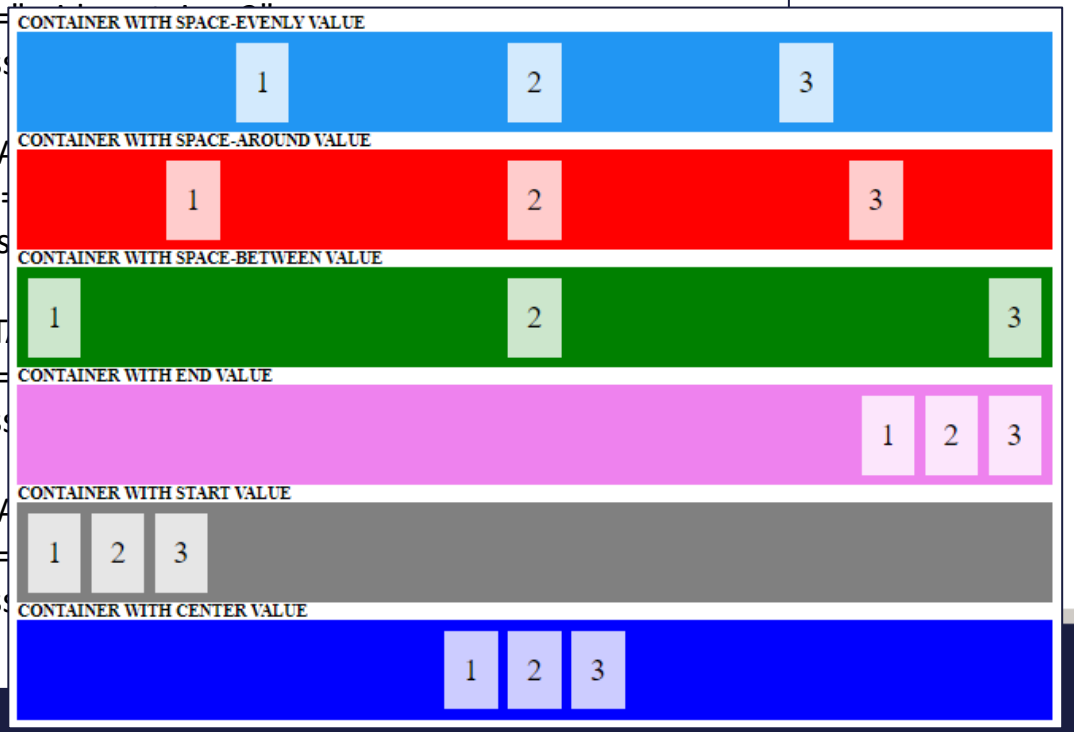
- **justify-content property:** used to align the entire grid within the container. It includes values such as:
  - **space-evenly**: It provides equal space in between or around the columns
  - **space-around:** It provides equal space around the columns
  - **space-between:** It gives an equal amount of space between the columns
  - **center**: It is used to align the grid in the middle of the container
  - **start**: It is used to align the grid at the beginning of the container
  - **end**: It is used to align the grid at the end of the container
- **align-content:** property used to align the entire grid within the container vertically
- Note: It is noted that the total height of the grid should be less than the height of the container for any effect of the align-content property
- The values of the align-content property are the same as the values of the justify-content property

justify-content example

AASTU

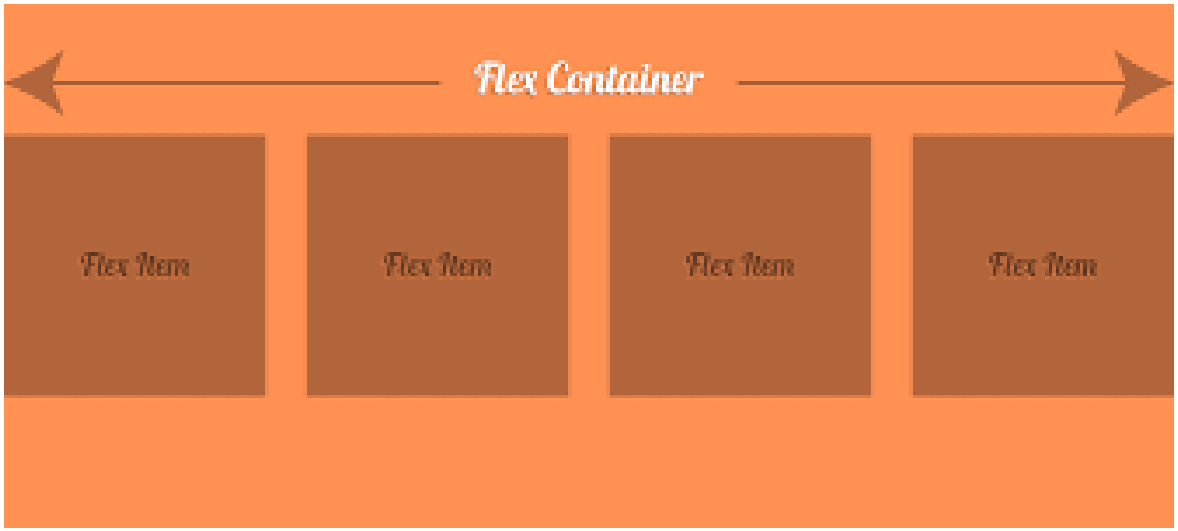
```
<style>
.grid-container1 {  display: grid;
  justify-content: space-evenly;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;…}
.grid-container2 {  display: grid;
  justify-content: space-around;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;…}
.grid-container3 {   display: grid;
  justify-content: space-between;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;…}
.grid-container4 {  display: grid;
  justify-content: end;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;…}
.grid-container5 {
  display: grid;
  justify-content: start;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;…}
.grid-container6 {  display: grid;
  justify-content: center;
  grid-template-columns: 50px 50px 50px;
  grid-gap: 10px;….}
.num {..
```

```
…..</head>
<body>
  <b>CONTAINER WITH SPACE-EVENLY VALUE</b>
<div class="grid-container1">
  <div class='num'>1</div>
  <div class='num'>2</div>
  <div class='num'>3</div>
</div>
<b>CONTAINER WITH SPACE-AROUND VALUE</b>
<div class="grid-container2">
  <div class='num'>1</div>
  <div class='num'>2</div>
  <div class='num'>3</div>
 </div>
 <b>CONTAINER WITH SPACE-BETWEEN VALUE</b>
<div class=
  <div class
 </div>
 <b>CONTA
 <div class=
   <div clas
 </div>
 <b>CONT
<div class=
  <div class
 </div>
 <b>CONTA
<div class=
  <div class
 </div>…
```



CSS Flexbox (Flexible boxes)

- The CSS3 flexbox is used to make the elements behave predictably when they are used with different screen sizes and different display device
- It is mainly used to make CSS3 capable to change its item's width and height to best fit for all available spaces
- The CSS3 flexbox contains flex containers and flex items
  - **Flex container**: the flex container specifies the properties of the parent. It is declared by setting the display property of an element to either flex or inline-flex
  - **Flex items**: the flex items specify properties of the children. There may be one or more flex items inside a flex container
- Everything outside a flex container and inside a flex item is considered as usual



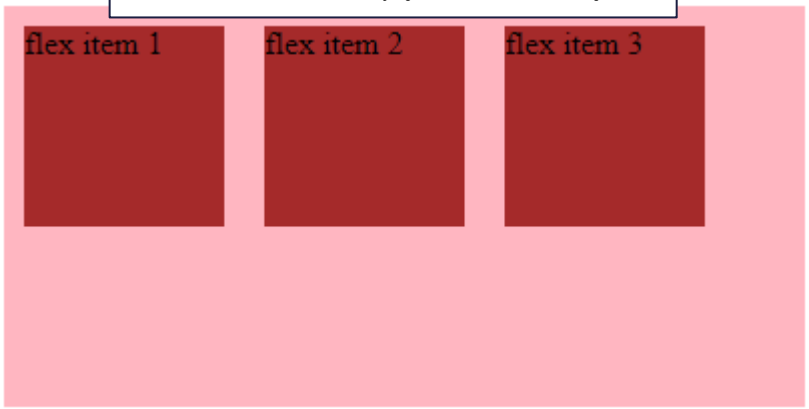
CSS3 Flexbox Properties

property	description
display	it is used to specify the type of box used for an html element.
flex-direction	it is used to specify the direction of the flexible items inside a flex container.
justify-content	it is used to align the flex items horizontally when the items do not use all available space on the main-axis.
align-items	it is used to align the flex items vertically when the items do not use all available space on the cross-axis.
flex-wrap	it specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line.
align-content	it is used to modify the behavior of the flex-wrap property. it is similar to align-items, but instead of aligning flex items, it aligns flex lines.
flex-flow	it specifies a shorthand property for flex-direction and flex-wrap.
order	it specifies the order of a flexible item relative to the rest of the flex items inside the same container.
align-self	it is used on flex items. it overrides the container's align-items property.
flex	it specifies the length of a flex item, relative to the rest of the flex items inside the same container.

```
..
.flex-container {
    display: -webkit-flex;
    direction: rtl;
    display: flex;
    width: 400px;
    height: 200px;
    background-color: lightpink;
}
.flex-item {
    background-color: brown;
    width: 100px;
    height: 100px;
    margin: 10px;
}
....
<body>
<div class="flex-container">
  <div class="flex-item">flex item 1</div>
  <div class="flex-item">flex item 2</div>
  <div class="flex-item">flex item 3</div>
</div>…
```

CSS3 Flexbox example

direction: rtl; /*defaul*/



direction: rtl;

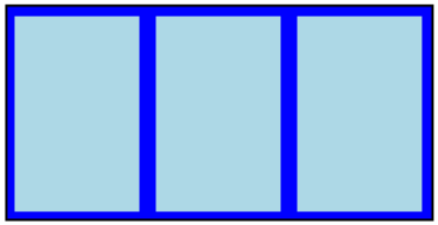
CSS flex property

- The flex property in CSS is shorthand for flex-grow, flex-shrink, and flex-basis. It only works on the flex-items
- flex property is used to set how a flex-item will shrink or grow to fit in the space
- The flex property can be specified by one, two, or three values
  - **one-value:** can be specified by a number or keywords such as none, auto, or initial
  - **two-value:** the first value must be a number (used as flex-grow), the second value can either be a number (used for flex-shrink) or a valid width value (used as flex-basis)
  - **three-value:** then the values must follow the order, a number for the flex-grow, a number for the flex-shrink, and a valid width value for flex-basis
- Syntax
  - flex: flex-grow flex-shrink flex-basis | auto | none | initial | inherit;
- **flex-grow**: specifies how much the item will grow compared to the other flexible-items
- **flex-shrink**: specifies how much the item will shrink compared to the other flexible-items
- **flex-basis:** It is used to set the length of the flex-item
- **auto**: This value of the flex property is equivalent to 1 1 auto
- **none**: This value of the flex property is equivalent to 0 0 auto

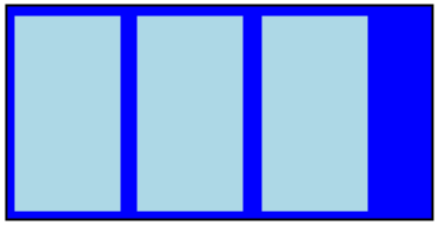
CSS flex example

```
.container { width: 200px;
height: 100px;
border: 1px solid black;
display: flex;
margin: 15px;
background-color: blue; }
.flex-item{ flex: 1; // unitless number: flex-grow }
.flex-item1{ flex: 0 50px; // flex-grow, flex-basis }
.flex-item2{ flex: 2 2; // flex-grow, flex-shrink }
.flex-item, .flex-item1, .flex-item2 {
background-color: lightblue;
margin: 4px; }
……
<h3> flex: 1; </h3>
<div class = "container">
  <div class = "flex-item"></div>
  <div class = "flex-item"></div>
  <div class = "flex-item"></div>
</div>
<h3> flex: 0 50px; </h3>
<div class = "container">
  <div class = "flex-item1"></div>
  <div class = "flex-item1"></div>
  <div class = "flex-item1"></div>
</div>
<h3> flex: 2 2; </h3>
<div class = "container">
<div class = "flex-item2"></div>
<div class = "flex-item2"></div>
<div class = "flex-item2"></div>
</div>…
```

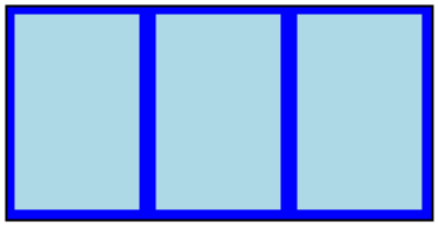
flex: 1;



flex: 0 50px;



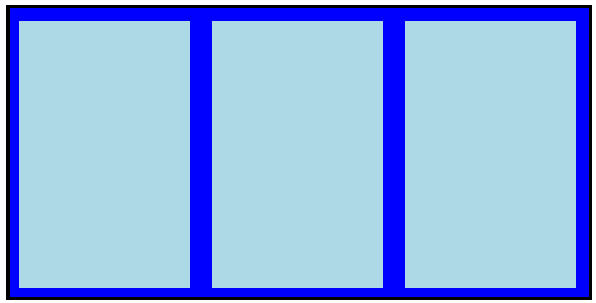
flex: 2 2;



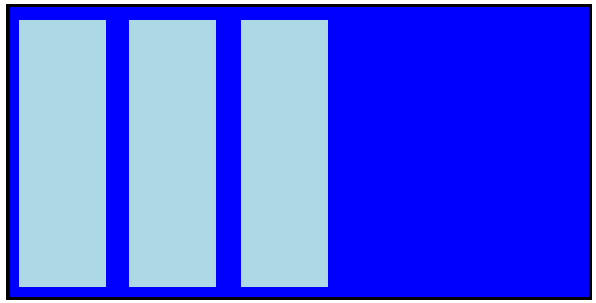
CSS flex example

```
.container { width: 200px;
height: 100px;
border: 1px solid black;
display: flex;
margin: 15px;
background-color: blue; }
.flex-item{ flex: auto; }
.flex-item1{ flex: 0 1 30px;}
.flex-item, .flex-item1{
background-color: lightblue;
margin: 4px; }
.....
<h3> flex: auto; </h3>
<div class = "container">
<div class = "flex-item"></div>
<div class = "flex-item"></div>
<div class = "flex-item"></div>
</div>
<h3> flex: 0 1 30px; </h3>
<div class = "container">
<div class = "flex-item1"></div>
<div class = "flex-item1"></div>
<div class = "flex-item1"></div>
</div>
```

flex: auto;



flex: 0 1 30px;



CSS flex-basis property

- Syntax
  - flex-basis: auto | width | initial | inherit;
- **auto**: It is the default value. This value sets the item's width equal to the value of its width property, if defined
- **width**: This value is defined using relative or absolute units

```
.container { display: flex; background-color: lightblue; }
.flex-item {
background-color: white;
text-align: center;
line-height: 40px;
font-size: 25px;
margin: 5px; }
</style>
……
<div class = "container">
<div class = "flex-item" style = "flex-basis: auto;"> auto </div>
<div class = "flex-item" style = "flex-basis: initial;"> initial </div>
<div class = "flex-item" style = "flex-basis: inherit;"> inherit </div>
 <div class = "flex-item" style = "flex-basis: 150px;"> 150px </div>
<div class = "flex-item" style = "flex-basis: auto"> auto </div>
</div>….
```

Example of the flex-basis property

auto	initial	inherit	150px	auto

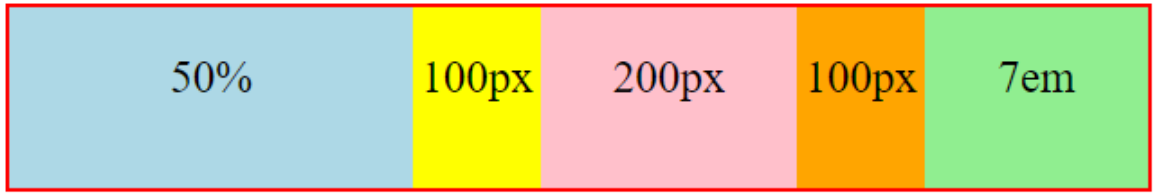
CSS flex-basis example

AASTU

```
.container { height: 100px;
border: 2px solid red;
display: flex;
background-color: blue; }
.container div{ padding-top: 25px; flex-basis: 100px; }
.flex-item{ text-align: center; font-size: 25px; }
.container div:nth-of-type(1) { flex-basis: 50%; }
.container div:nth-of-type(3) {flex-basis: 200px; }
.container div:nth-of-type(5) { flex-basis: 7em; }
.....
<h1>The flex-basis Property</h1>

<div class = "container">
<div class = "flex-item" style= "back
lightblue;"> 50%</div>
<div class = "flex-item" style= "background-color:
yellow;"> 100px</div>
<div class = "flex-item" style= "background-color:
pink;"> 200px </div>
<div class = "flex-item" style= "background-color:
orange;">100px </div>
<div class = "flex-item" style= "background-color:
lightgreen;">7em</div>
</div>..
```

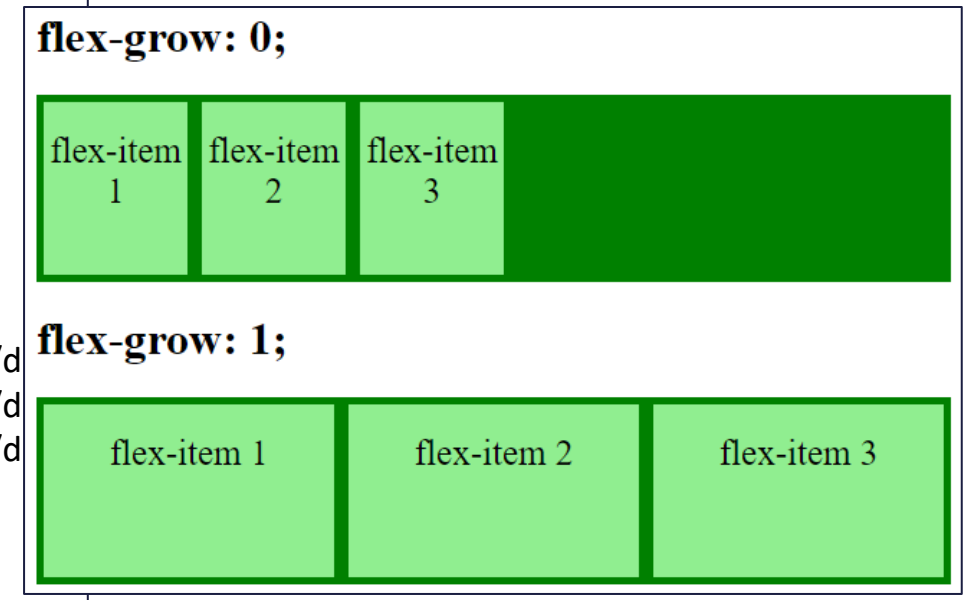
The flex-basis Property



CSS flex-grow property

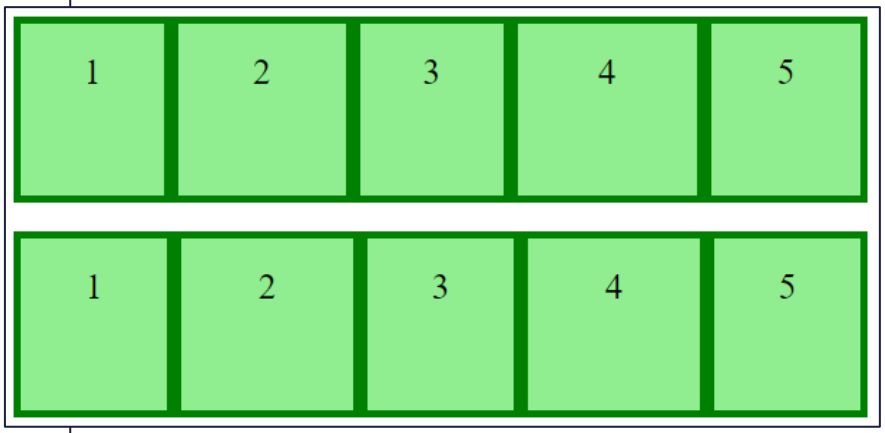
- The CSS flex-grow property specifies how much space a flex item should take if there is available space
  - Example
    - If we set flex-grow to 1 for all items, each item will set to an equal size in the container
    - If we give any of the items a value of 2, then the corresponding item will take up space twice than the others
- Syntaxflex-grow: number| initial | inherit;

```
.container { display: -webkit-flex; display: flex;
background-color: green; }
.flex-item { background-color: lightgreen;
text-align: center; font-size: 25px;
width: 100px; height: 100px;
padding-top: 20px; margin: 5px; }
....
<h1> flex-grow: 0; </h1>
<div class="container">
<div class="flex-item" style = "flex-grow: 0;"> flex-item 1 </d
<div class="flex-item" style = "flex-grow: 0;"> flex-item 2 </d
<div class="flex-item" style = "flex-grow: 0;"> flex-item 3 </d
</div>
<h1> flex-grow: 1; </h1>
<div class="container">
<div class="flex-item" style = "flex-grow: 1;"> flex-item 1 </div>
<div class="flex-item" style = "flex-grow: 1;"> flex-item 2 </div>
<div class="flex-item" style = "flex-grow: 1;"> flex-item 3 </div>
</div>
</div>…
```



CSS flex-grow example

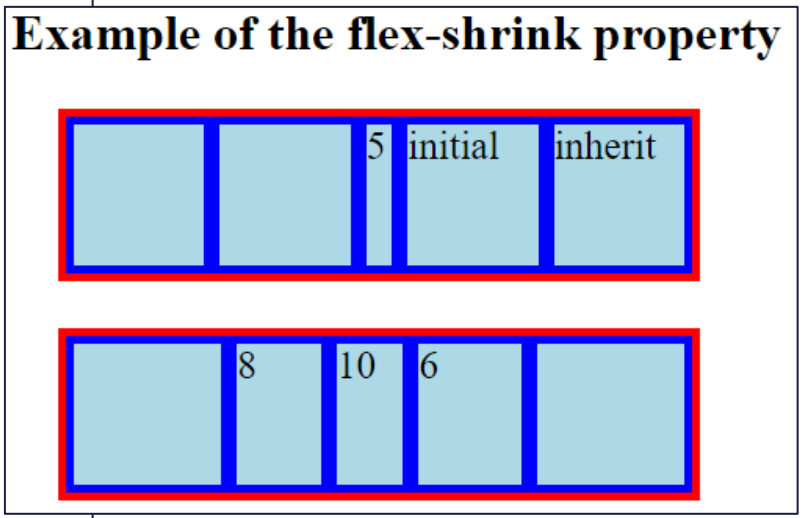
```
.container { display: -webkit-flex; display: flex;
background-color: green; margin: 20px; }
.flex-item {
background-color: lightgreen;
text-align: center;
font-size: 25px;
width: 100px; height: 100px;
padding-top: 20px; margin: 5px;
}
…
<div class="container">
<div class="flex-item" style = "flex-grow: 0;"> 1 </div>
<div class="flex-item" style = "flex-grow: 4;"> 2 </div>
<div class="flex-item" style = "flex-grow: 0;"> 3 </div>
<div class="flex-item" style = "flex-grow: 6;"> 4 </div>
<div class="flex-item" style = "flex-grow: 1;"> 5 </div>
</div>
<div class="container">
<div class="flex-item" style = "flex-grow: 1;"> 1 </div>
<div class="flex-item" style = "flex-grow: 9;"> 2 </div>
<div class="flex-item" style = "flex-grow: 1;"> 3 </div>
<div class="flex-item" style = "flex-grow: 9;"> 4 </div>
<div class="flex-item" style = "flex-grow: 1;"> 5 </div>
</div>…
```



CSS flex-shrink property

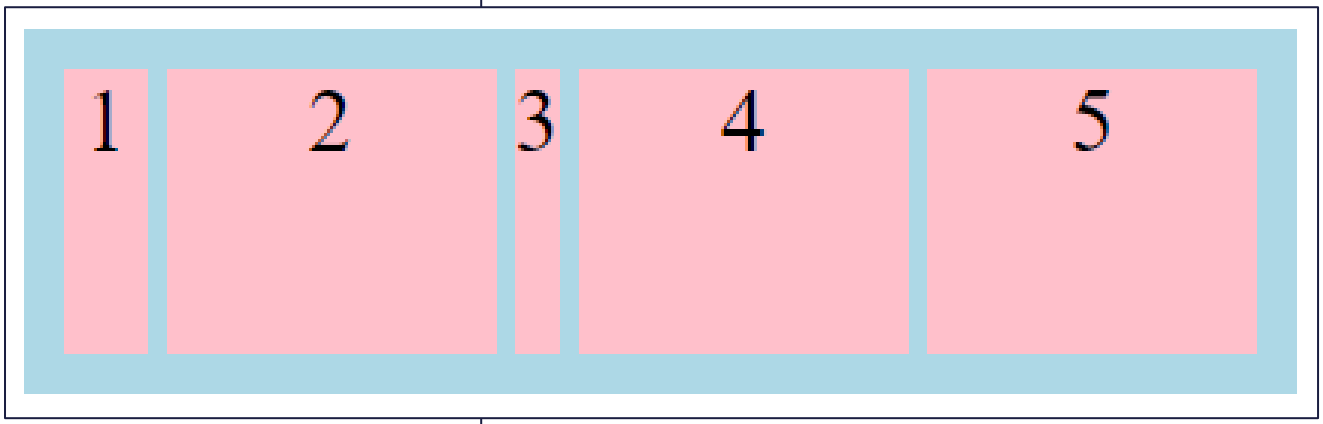
- The CSS flex-shrink property specifies how much an item will shrink than the other items of the container
- Example
  - The flex-item with the flex-shrink: 2; will shrink twice than the flex-shrink: 1;
  - The higher the flex-shrink value causes the item to shrink more than the others.
- **Syntax** - flex-shrink: number| initial | inherit;

```
.container { width: 400px; height: 100px;
border: 5px solid red; display: flex; margin: 30px;
background-color: blue; }
.flex-item{  background-color: lightblue;
font-size: 25px; margin: 5px;
flex-grow: 1; flex-shrink: 1;  flex-basis: 100px; } …
<div class="container">
<div class = "flex-item"></div>
<div class = "flex-item"></div>
<div class = "flex-item" style = "flex-shrink: 5;"> 5 </div>
<div class = "flex-item" style = "flex-shrink: initial;"> initial </div>
<div class = "flex-item" style = "flex-shrink: inherit;"> inherit </div>
</div>
<div class="container">
<div class = "flex-item"></div>
<div class = "flex-item" style = "flex-shrink: 8;"> 8 </div>
<div class = "flex-item" style = "flex-shrink: 10;"> 10 </div>
<div class = "flex-item" style = "flex-shrink: 6;"> 6 </div>
<div class = "flex-item"></div>
</div>…..
```



CSS flex-shrink example

```
.container {
width: 400px; height: 100px;
padding: 10px; display: flex;
background-color: lightblue;
}
.flex-item {
background-color: pink;
margin: 3px;
text-align: center;
font-size: 30px;
flex-basis: 150px;
flex-grow: 0;
}
....
<div class="container">
<div class="flex-item" style = "flex-shrink: 3;">
1 </div>
<div class="flex-item" > 2 </div>
<div class="flex-item" style = "flex-shrink: 5;" >
3 </div>
<div class="flex-item" > 4 </div>
<div class="flex-item" > 5 </div>
</div>..
```



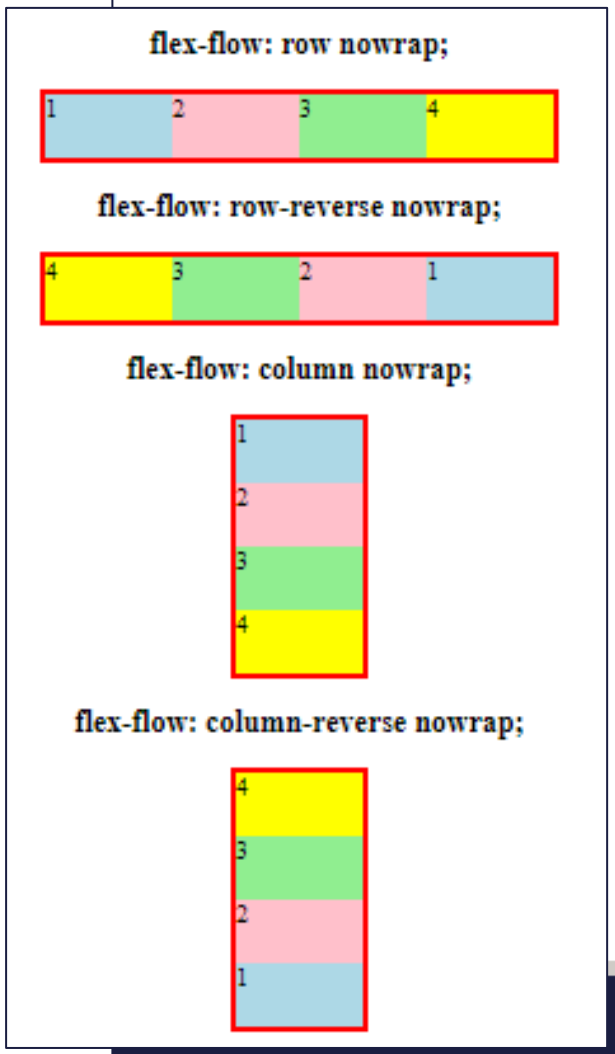
CSS flex-flow property

- This CSS property is shorthand for **flex-direction** and **flex-wrap properties**
- Syntax
  - flex-flow: flex-direction flex-wrap | initial | inherit;

Values	Description
flex-direction	The flex-direction property is used to set the direction of the flexible items inside the flex container. Its default value is **row** (left-to-right, top-to-bottom). The possible values of this property are **row, row-reverse, column**, and **column-reverse**.
flex-wrap	The flex-wrap property specifies if the flex-items should wrap or not, in case of not enough space for them on one flex line. Its default value is **nowrap**. The possible values of this property are **nowrap, wrap**, and **wrap-reverse**.

CSS flex-flow example

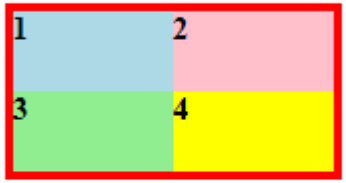
```
.mainrow{ width: 400px; height: 50px; border: 5px solid red; }
.maincol{ width: 100px; height: 200px; border: 5px solid red; }
#row { flex-flow: row nowrap; }
#rowrev { flex-flow: row-reverse nowrap; }
#col { flex-flow: column nowrap; }
#colrev { flex-flow: column-reverse nowrap; }
div { width: 100px; height: 50px; display: flex; font-size: 20px; }
….. <h2>  flex-flow: row nowrap;</h2>
<div id= "row" class = "mainrow">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div>  <h2>  flex-flow: row-reverse nowrap;</h2>
<div id= "rowrev" class = "mainrow" >
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div>   <h2>  flex-flow: column nowrap;</h2>
<div id="col" class = "maincol">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div>   <h2>  flex-flow: column-reverse nowrap;</h2>
<div id="colrev" class = "maincol">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div> </div>
```



```
.mainrow{ width: 200px; height: 100px; border: 5px solid red; }
.maincol{ width: 200px; height: 150px; text-align: left; border: 5px
solid red; }
#row { flex-flow: row wrap; }
#rowrev { flex-flow: row-reverse wrap; }
#col { flex-flow: column wrap; }
#colrev { flex-flow: column-reverse wrap; }
div { width: 100px; height: 50px; display: flex;
font-size: 20px; font-weight: bold; }
… <h2>  flex-flow: row wrap;</h2>
<div id= "row" class = "mainrow">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div> <h2>  flex-flow: row-reverse wrap;</h2>
<div id= "rowrev" class = "mainrow" >
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div> <h2>  flex-flow: column wrap;</h2>
<div id="col" class = "maincol">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div> <h2>  flex-flow: column-reverse wrap;</h2>
<div id="colrev" class = "maincol">
<div style="background-color: lightblue;"> 1 </div>….
```

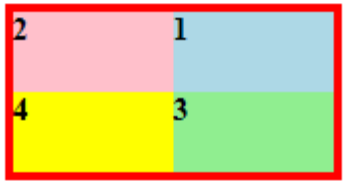
CSS flex-flow example

flex-flow: row wrap;

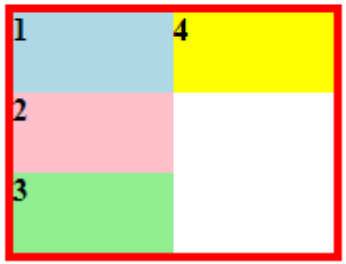


using **wrap**

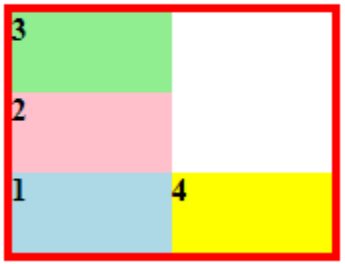
flex-flow: row-reverse wrap;



flex-flow: column wrap;



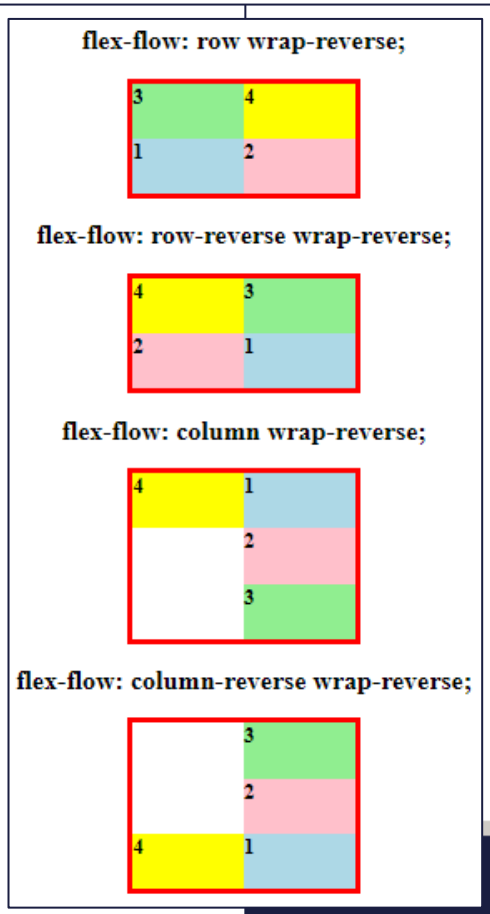
flex-flow: column-reverse wrap;



CSS flex-flow example

```
.mainrow{ width: 200px; height: 100px; border: 5px solid red;}
.maincol{ width: 200px; height: 150px; text-align: left; border: 5px solid red; }
#row { flex-flow: row wrap-reverse; }
#rowrev { flex-flow: row-reverse wrap-reverse; }
#col { flex-flow: column wrap-reverse; }
#colrev { flex-flow: column-reverse wrap-reverse; }
div { width: 100px; height: 50px; display: flex; font-size: 20px; font-weight:
bold; }
…….<h2>  flex-flow: row wrap-reverse;</h2>
<div id= "row" class = "mainrow">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div> <h2>  flex-flow: row-reverse wrap-reverse;</h2>
<div id= "rowrev" class = "mainrow" >
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div> <h2>  flex-flow: column wrap-reverse;</h2>
<div id="col" class = "maincol">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>
<div style="background-color: yellow;"> 4 </div>
</div> <h2>  flex-flow: column-reverse wrap-reverse;</h2>
<div id="colrev" class = "maincol">
<div style="background-color: lightblue;"> 1 </div>
<div style="background-color: pink;"> 2 </div>
<div style="background-color: lightgreen"> 3 </div>..
```

Using **wrap-reverse** value of the **flex-wrap** & all the possible values of the **flex-direction** property



Reading Assignment

- Animation
- Translate
- Transition
- Pseudo-classes
- Pseudo-elements
- Gradients
- Masking
- @keyframes
- Units
- Combinators
- Minify
- 2D Transforms
- 3D Transforms

CSS media queries

- Media queries allow you to apply CSS styles depending on a **device's general type** (such as print vs. screen) or other characteristics such as **screen resolution** or **browser viewport width**
- Media queries are used for the following:
  - To conditionally apply styles with the CSS @media and @import at-rules.
  - To target specific media for the <style>, <link>, <source>, and other HTML elements with the media= attribute.
  - To test and monitor media states using the Window.matchMedia() and MediaQueryList.addListener() JavaScript methods
- A media query is composed of an optional **media type** and **any number of media feature expressions**
- **Media types** define the broad category of device for which the media query applies: all, print, screen. The type is optional (assumed to be all) except when using the not or only logical operators.
- **Media features** describe a specific characteristic of the user agent, output device, or environment
  - E.g. any-hover, any-pointer, aspect-ratio, color, display-mode, height, width…
- For example, the hover feature allows a query to test against whether the device supports hovering over elements
- **Logical operators** can be used to compose a complex media query: **not**, **and**, **or** and **only**

CSS Media Query syntax/example

- **Targeting media types:**
  - @media print { /* … */}
  - You can also target multiple devices
    - @media screen, print { /* … */ }
- **Targeting media features:**
  - @media (hover: hover) { /* … */}
- Many media features are range features, which means they can be prefixed with "min-" or "max-" to express "minimum condition" or "maximum condition" constraints
  - @media (max-width: 1250px) { /* … */}
- **Combining multiple types or features:**
  - @media (min-width: 30em) and (orientation: landscape) { /* … */ }
  - To limit the styles to devices with a screen, you can chain the media features to the screen media type:
  - @media screen and (min-width: 30em) and (orientation: landscape) { /* … */}
- **Testing for multiple queries:** you can use a comma-separated list to apply styles when the user's device matches any one of various media types, features, or states
  - @media (min-height: 680px), screen and (orientation: portrait) { /* … */ }

CSS Media Query syntax/example

- **Inverting a query's meaning**
  - @media not all and (monochrome) { /* … */ }
  - This means that the above query is evaluated like this:
    - @media not (all and (monochrome)) { /* … */ }
  - It wouldn't be evaluated like this:
    - @media (not all) and (monochrome) { /* … */ }
  - As another example, the following media query:
    - @media not screen and (color), print and (color) { /* … */ }
  - This means that the above query is evaluated like this:
    - @media (not (screen and (color))), print and (color) { /* … */ }
- The **only** keyword prevents older browsers that do not support media queries with media features from applying the given styles
  - It has no effect on modern browsers
  - @media only screen and (color) { /* … */ }

CSS Media Query Syntax in Level 4

- @media (max-width: 30em) { /* … */ } in level 4 **@media (width <= 30em) { /* … */ }**
- @media (min-width: 30em) and (max-width: 50em) { /* … */ } in level 4 **@media (30em <= width <= 50em) { /* … */}**
- **Negating a feature with not:** Using not() around a media feature negates that feature in the query
  - @media (not(hover)) { /* not(hover) would match if the device had no hover capability*/}
- Testing for multiple features with **or**
  - @media (not (color)) or (hover) { /* … */ }

Exercise

- We can use CSS layout to design web pages
  - There are 3 ways to design layout of a web page:
  - HTML Div with CSS: fast and widely used now
  - HTML Table: slow and less preferred
  - HTML Frameset: deprecated now
- Loader
- Tooltips
- Arrow
- Aural Media
- User Interface
- Pagination