Reference links

|  |  |
| --- | --- |
| Book – for dummies GIT hub repo links | <https://github.com/paulmcfe/web-coding-and-dev-fd> |
|  | <https://webdevworkshop.io/code/> |
| All color names | <https://www.w3schools.com/tags/ref_colornames.asp> |
| To use google fonts | <https://fonts.google.com/icons> |
| Nice font url | <https://fonts.google.com/specimen/Nerko+One> |
| API docs for developers – it has all fonts will all effect | <https://developers.google.com/fonts> |
|  |  |

About java script

It is to get the action, like onclick()

Tag attribute element

<input type= “text”/> enter ur pass here <input>

Here input is the tag, which ever opening and closing bracket

Type is attribute

Total line is called element = Tag+ data inside that

How to apply Styles

1. Inline – this is the worst approach bec if we want to apply same style for diff buttons/tags we should do copy paste

Means we should write styles in every tag in style attribute, in below paragraph tag we wrote in style attribute

<h1 style="background-color: black;color: darkseagreen;">Lead position is hectic</h1>

<p style="background-color: blueviolet;color: aliceblue;">i cant take extra load</p>

    <p style="background-color:yellowgreen;color: aliceblue;">i dont want lead position</p>

1. Internal style sheets

Means applying styles in same html page under html tag as below – this is also bad approach, bec if we want same styles in another html we should go for copy paste

// Here we used tag selector, here p is tag selector, we can use id selector also

<head>

    <title>Document</title>

    <style>

            p{

                background-color: deeppink;

                color: whitesmoke;

                font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

            }

            h1{

                background-color: violet;

                color: aliceblue;

            }

    </style>

</head>

<body>

    <h1 >Lead position is hectic</h1>

    <p >i cant take extra load</p>

    <p >i dont want lead position</p>

</body>

</html>

1. External style sheets

Styles should be in separate file with .css extension and link to a html file using below tag

rel =”stylesheet” means we are relating to a style sheet

if css file and current html is in same folder we can just mention css file name

|  |  |
| --- | --- |
| <!DOCTYPE html>  <head>      <link rel="stylesheet" href="s1.css">  </head>  <body>      <h1 >Lead position is hectic</h1>      <p >i cant take extra load</p>      <p >i dont want lead position</p>    </body>  </html> | S1.css content  // Here p is called tag selector  p{      background-color: slateblue;      color: aliceblue;  }  h1{      background-color: peru;      color: papayawhip;  } |

Important note

1. Note:- the properties like (top,left will be applicable only when u keep position: relative/ fixed/absolute) ..anything other than static
2. In myntra website all images size both width & height depends on viewing height and width, if screen size/tab area is reduced, image area is also getting small

If screen size is increases total image also should increase

Properties of element

|  |  |
| --- | --- |
| Background-color for single color  background-color: salmon;  Use property background for – gradient/mixed background colours   background: radial-gradient(          rgb(190, 231, 41) 10%,          rgb(225, 59, 231) 20%,          #2f5566 20%        );  Transitions:- changes will be applied slowly on hover / on any action instead of applying immediately  transition: <property name> <seconds to transform> | Font related  Font-size:90px – to increase font size  Font-weight :bolder—  For gradient text  Font-family //to choose a style  Text-align:left  Opacity: 0.6 means bit transperant |
| Height  Ex:-   h3 {        background-color: darkmagenta;        color: white;        height: 500px;  border-radius:50% -- means circular shape      }   <h3>See all the images</h3>  Radius means distance from center of circle to any point of boundary | Height means it is the height(vertical ||) of that element    Here as u gave 500px height, the height of the elements was considered as 500px  Default height will be taken based on content- single line or full line based on content height |
| Width (----) addam  Horizontal Width of that particular element  There is no property called length | By default – width will be taken 100% horizontal full line entire screen |
| Top   #t3 {        height: 1000px;        top: 25px;        background-color:  turquoise;        position: relative;      }      <h4 id="t3">Ladies wear</h4> | top means - space on above of that element    Here for <h4 id="t3">Ladies wear</h4> we gave top as 25 px , so it got 25px space above it |
| font-size  html {  **font-size: 16px;**      }      #t1 {        background-color: yellowgreen;        font-size: 2rem;        /\* by default 1 rem root ele font = 16px        so here final font size will be taken as  =32 px \*/        width: 10em;        /\* as current font size =32px then width here is 320px \*/      }      #px320 {  **background-color: pink;**        width: 320px;      } | <div id="t1">Charan is lazy fellow</div>      <div id="px320">Santhoshi is lazy fellow</div> |
| Border-box      #cb {  **box-sizing: content-box;**        font-size: 2rem;        background-color: salmon;        width: 400px;        padding: 10px;        border: 20px dashed royalblue;        height: 100px;      } |  |
|  |  |

Tips

1. To move from left and right – use translateX() function

Selectors

Using selectors we will specify to which content we can apply styles

|  |  |
| --- | --- |
| Simple selectors | select elements based on tag name, id (id should be unique for each element), class (as many people belong to same class, similarly many tags can belong to same class) |
| Combinatory | Tag+ class . tag+ id |
| Pseudo class selectors | Applying styles on Onhover() on button , on hover() on anchor tag..  select elements based on a certain state |
| Pseudo element selectors | select and style a part of an element |
| Attribute selectors | <input type= “text”>  <input type= “email”> select elements based on an attribute or attribute value  See here based on attribute values only we can apply styles |

Simple selectors

|  |  |  |
| --- | --- | --- |
| /\* this is called tag selector \*/          h2{              background-color: turquoise;              color: peru;          } | /\* # is the id selector \*/  Id will be unique aswell as hashtag will also be unique          #tcs{              background-color: magenta;              color: white;          } | /\* Dot. represents class selector \*/          .greener{              background-color: goldenrod;          }   .child {          width: 200px;          height: 100px;        }        .c1 {          background-color: aqua;        }  <div class="c1 child">one</div>  For One tag we can apply many classes  Like how a single student belongs to multiple classes  We can apply multiple classes for 1 tag |

Combinator selector

* descendant selector (space)
* child selector (>)
* adjacent sibling selector (+)
* general sibling selector (~)

|  |
| --- |
| <style>          div >h2 {              background-color: gold;          }          h3+h4{              background-color: mediumslateblue;          }      </style> |

Pseudo element selector

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [::after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Insert something after the content of each <p> element  Note:- here instead of tag selector u can happily use id or class selector also |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | #f1::before | Insert something before the content of each id f1 element |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | Selects the first letter of each <p> element |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | Selects the first line of each <p> element |
| [::marker](https://www.w3schools.com/cssref/sel_marker.asp) | ::marker | Selects the markers of list items |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | p::selection | Selects the portion of an element that is selected by a user |

Examples

|  |  |
| --- | --- |
| <head>  <style>          /\* here we are using tag selector \*/          h2::before{              content: "hello ap bye bye ycp";              color: chocolate;              font-size: xx-large;          }          /\* here we are using id selector \*/          #tcs::after{              content: "jai TDP";              color: goldenrod;              font-size: xx-large;          }          /\* here . means we are using class selector \*/          .greener::first-letter{                  font-size: xx-large;                  color: yellowgreen;          }          /\* here i have used combinator selector  \*/          div > p::selection{              color: yellowgreen;              background-color: darkgrey;          }      </style>  </head> | <body>      <h2>tcs is having wf projects</h2>      <div>          <h2>synergy park</h2>      </div>      <h3 class="greener">tcs is in all countries</h3>      <h4  class="greener">wells fargo in in u.s.a</h4>      <h3 id="tcs">tcs is in india and usa also</h3>      <h3 id="tcs">tcs is in india and usa also</h3>      <h4  class="greener">sf is in illinois</h4>      <p class="greener">i cant take extra load</p>      <p >i dont want lead position</p>      <div>          <p >i want </p>      </div>  </body> |

Attribute selector

<input type= “text”>

<input type= “email”> select elements based on an attribute or attribute value

See here based on attribute values only we can apply styles ,

Ex:- if attribute value=email then only do this

|  |  |  |
| --- | --- | --- |
| **Selector** | **Example** | **Example description** |
| [[*attribute*]](https://www.w3schools.com/cssref/sel_attribute.php) | [target] | Selects all elements with a target attribute |
| [[*attribute*=*value*]](https://www.w3schools.com/cssref/sel_attribute_value.php) | [target="\_blank"] | Exact match  Selects all elements with target="\_blank" |
| [[*attribute*~=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_contains.php) | [title~="flower"] | ~ means contains  Selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower" |
| [[*attribute*|=*value*]](https://www.w3schools.com/cssref/sel_attribute_value_lang.php) | [lang|="en"] | Selects all elements with a lang attribute value starting with "en" |
| [[*attribute*^=*value*]](https://www.w3schools.com/cssref/sel_attr_begin.php) | a[href^="https"] | Selects all <a> elements with a href attribute value **starting with** "https" |
| [[*attribute*$=*value*]](https://www.w3schools.com/cssref/sel_attr_end.php) | a[href$=".pdf"] | Selects all <a> elements with a href attribute value **ending with** ".pdf" |
| [[*attribute*\*=*value*]](https://www.w3schools.com/cssref/sel_attr_contain.php) | a[href\*="w3schools"] | Selects all <a> elements with a href attribute value **containing the substring** "w3schools" |

|  |  |
| --- | --- |
| <style>      /\* This = is called exact match  \*/  /\* syntax is tag[attribute=value] \*/              input[type="text"]{                      background-color: yellowgreen;              }        /\* This = is called exact match  \*/      input[type="email"]{          background-color: teal;      }      /\* This is demo for start with attribute -- \*/      p[title^="tata"]{          color: yellowgreen;          font-size: larger;      }      /\* This is demo for contains attribute \*/      h2[title\*="technology"]{          background-color: darkorange;      }  </style> | <body>      enter ur user name <input type="text"> <br>      enter ur password <input type="email"> <br>        <p title="tata consultancy services">tcs is leader in gen ai</p>      <p title="tataconsultancy services">tcs is leader in gen ai</p>      <p title="tataconsultancyservices">tcs is leader in gen ai</p>      <h2 title="cognizant technology solutions"> CTS</h2>      <h2 title="cognizanttechnologysolutions"> CTS</h2>      <h2 title="cognizant technology solutions"> CTS</h2>  </body>  </html> |

Pseudo class selector

Selects elements based on certain state like hovering on paragraph

* Style an element when a user mouses over it
* Style visited and unvisited links differently
* Style an element when it gets focus

|  |  |
| --- | --- |
| selector:pseudo-class name {   property: value; } | Note here we can happily use id or tag selector |

ex:- pseudo classes – link(un visited), visited (already clicked), readonly, readwrite,

active (when u just clicked – means for that sec that will be in active state)- this is for button, anchor links..

target – target is a pseudo class where it will be activated when someone clicked that a href tag

hover – this pseudo class is applicable to all elements including paragraph

Focus:- in a form when user is entering some value in a text box or if that user mouse over on that field then that field will be in focus state

By default link will be in unvisited state

Below 4 will be applicable for form fields like input

<input type=”text” required=true readonly minlength=”5” >

//this is required, when required =false then it will be in optional state

//

input:focus{} //Here input is the tag name // here focus is a state

input:required{}

input:optional{}

input:read-only{} //read only is an another pseudo class which we can’t enter any input, if read only is not there,

by default its read write

input:valid{} // this is a pseudo class where if minlenth=5, if we gave 4 chars then it will be in invalid state

input:invalid{}// if conditions are not satisfied then it will be in invalid state

input:disabled{}// this is used only for buttons <input type= “text” disabled>

like disabling a button, if validations are not met then disable the button

p:first-child{}

means if paragraph is the first child of any div tag or body tag then apply this properties

then problem is if p is first child then only apply this, but if paragraph is second element this wont work

p:last-child{}

p:nth-child(2){}

|  |  |  |
| --- | --- | --- |
| p:first-child{} |  | input:first-of-type  so if u want to apply some properties to first paragraph then we have to use first-of-type even if before this paragraph if some input tag is there also then also no problem |
|  |  |  |

**p:last-of-type** : means **apply to last paragraph** (even if some elements are there before that also ok)

**p:last-child**{} this means **if paragraph is the last child** then apply these last properties

but make sure after this no element must be there

Here this selector can be simple selector (tag or id or class) or combinator selector

|  |  |  |
| --- | --- | --- |
| **Selector/pseudo classes** | **Example** | **Example description** |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | Selects the active link |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | Selects every checked <input> element |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | Selects every disabled <input> element |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | Selects every <p> element that has no children |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | Selects every enabled <input> element |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | Selects every <p> elements that is the first child of its parent |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | Selects every <p> element that is the first <p> element of its parent |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Selects the <input> element that has focus |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | Selects links on mouse over |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | Selects <input> elements with a value within a specified range |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | Selects all <input> elements with an invalid value |
| [:lang(*language*)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | Selects every <p> element with a lang attribute value starting with "it" |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | Selects every <p> elements that is the last child of its parent |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | Selects every <p> element that is the last <p> element of its parent |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | Selects all unvisited links |
| [:not(selector)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | Selects every element that is not a <p> element |
| [:nth-child(n)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | Selects every <p> element that is the second child of its parent |
| [:nth-last-child(n)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | Selects every <p> element that is the second child of its parent, counting from the last child |
| [:nth-last-of-type(n)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | Selects every <p> element that is the second <p> element of its parent, counting from the last child |
| [:nth-of-type(n)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | Selects every <p> element that is the second <p> element of its parent |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | Selects every <p> element that is the only <p> element of its parent |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | Selects every <p> element that is the only child of its parent |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | Selects <input> elements with no "required" attribute |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | Selects <input> elements with a value outside a specified range |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | Selects <input> elements with a "readonly" attribute specified |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | Selects <input> elements with no "readonly" attribute |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | Selects <input> elements with a "required" attribute specified |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | root | Selects the document's root element |
| [:target](https://www.w3schools.com/cssref/sel_target.asp) | #news:target | Selects the current active #news element (clicked on a URL containing that anchor name) |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | Selects all <input> elements with a valid value |
| [:visited](https://www.w3schools.com/cssref/sel_visited.asp) | a:visited | Selects all visited links |

a:link{

            color: hotpink;

            font-size: xx-large;

        }

        /\* 'Visited' is another pseudo class means we already clicked that anchor link \*/

        a:visited{

                color: yellowgreen;

        }

        /\* 'active' is another pseudo class when we click link for that sec that will be in active state \*/

        a:active{

            color: tomato;

        }

        /\* 'hover' is another pseudo class selector   \*/

        #p1:hover{

            color: violet;

            background-color: aliceblue;

            font-size: xx-large;

            font-family: 'Gill Sans', 'Gill Sans MT', Calibri, 'Trebuchet MS', sans-serif;

        }

        /\* Target is an another pseudo class, this will be active when anchor tag reached that state \*/

        #end:target{

            background-color: darkorange;

        }

        /\* focus is a another psuedo class when u click on text box it will hover \*/

        #uid:focus{

            background-color: darkorange;

            color: whitesmoke;

            box-sizing: border-box;

        }

        /\* invalid is a pseudo class when validation failed this class will come into exist \*/

        #uid:invalid{

            background-color: olivedrab

        }

        /\* if input tag contains disabled then this psuedo clas will be applied \*/

        #disa:disabled{

            background-color: crimson;

        }

        /\* if input tag contains required then it is called mandatory tag \*/

        #eid:required{

            background-color: tomato;

        }

        /\* if required tag is not there then it is called optional \*/

        #add:optional{

            background-color: yellow;

        }

        /\* checked is an another pseudo class this will be active when it is checked that checked beside label will be selected \*/

        input:checked{

            background-color: black;

        }

        /\* here + means immediate sibling \*/

        input:checked + label{

            /\* background-color: darkviolet; \*/

            font-style: italic;

            font-size: xx-large;

        }

        /\* means if paragraph is the first child of any div tag or body tag then apply this properties\*/

        p:first-child{

                background-color: palegreen;

                font-size: xx-large;

        }

        /\* If input tag is the 2nd child of any div tag or any tag then apply these properties \*/

        input:nth-child(3){

            background-color: plum;

        }

        /\* apply this properties for the input tag first of its type  \*/

        input:first-of-type{

            background-color: yellowgreen;

        }

        /\* apply this property for last paragraph \*/

        p:last-of-type{

            background-color:violet

        }

            /\* apply these properties if achor tag is the last child of any tag even body tag \*/

        div:last-child{

            background-color: crimson;

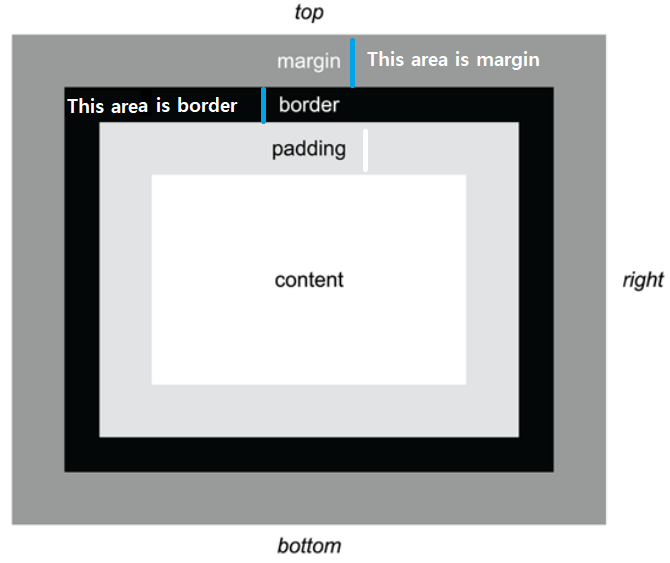
        }

Sample code

Box model

All our content will be in a box

ex:- <p> , <a href=”” > , <div> .. all this data will be inside a box



MBPC – Mahesh (margin) babu (border) picinary(padding) vasdu (content)

Padding is the space between content and the thick border (pg= padding means gap)

Margin is the space that is present between 2 element borders

|  |  |  |
| --- | --- | --- |
| For every box, we can apply 3 properties – margin, border , padding (space between margin and content)  Ex:- all block level elements (h2,h3), some inline elements (<div>,<p>)  Note:- For some inline elements we can’t apply margin- top,bottom is not applicable  Ex:- for <div> <i> | | |
| Padding: 15px 60px  Means 1st value = top & bottom  2nd value = left & right | Padding: 20px  Means all side 20px – top, bottom, left,right | Padding: 10px 20px 30px 40px  Means – clockwise  Top, right, bottom, left |
| Ex: border should have 3 properties  border: dashed 10px violet  border: style size colour  Style:- solid / dashed /dotted /double  Size: - 10px… (this is the width of the border) | Border 🡪 means this will be same all side  border-left 🡪 only for left  border-left: solid 10px blue  border-colour  border-right |  |
| Margin: 20px means all sides same margin | Margin-left: 20px only left margin is 20px |  |
|  |  |  |

So total width will be content width + padding width+ border width

We have to give all these widths individually

|  |  |
| --- | --- |
| <head>      <meta charset="UTF-8" />      <meta name="viewport" content="width=device-width, initial-scale=1.0" />      <title>Document</title>      <style>        h3 {          width: 300px;          /\* This width represents the width of that element \*/          background-color: lightgoldenrodyellow;          border: dashed 5px tomato;          padding: 30px 50px 30px 100px;          /\* top right bottom left \*/          margin-left: 50px;          margin-bottom: 0px;        }        h2 {          width: 300px;          background-color: tan;          border: solid 5px darkblue;          padding: 10px 100px 5px 30px;          margin-top: 10px;        }      </style>    </head>    <body>      <h3>Sambar in Plastic cover becoming cancerous</h3>      <h2>Govt must take an action</h2>    </body> | Some gap must be there na between border & content ,  That gap is called padding  margin is the Gap between 2 elements is called |
| Note:- For some inline elements we can’t apply margin- top,bottom is not applicable  Ex:- for <div> <i>  Good note is – we can apply at least apply right and left margins as they will be in separate box  Hence instead of applying margin top , bottom for those inline elements  U can apply those styles for next block elements |  |

Css position

For every element we have a position property

For position property we can give these 5 values – static, absolute, relative, fixed, sticky, inherit

|  |  |
| --- | --- |
| Static | It is the default position |
| Absolute (AP) | AP ~~ absolute means parent  It will take the reference as parent , if no parent found then, it will consider the body tag as the parent tag |
| Relative | RA- relative means actual position  Means, it will take reference from that actual desired position |
| Fixed | If position Is fixed means content will be in fixed state, even if u scroll the screen also it wont change  Ex:- like amazon nav bar |
|  |  |

|  |  |
| --- | --- |
| Note:- the properties like (top,left will be applicable only when u keep position: relative/ fixed/absolute) ..anything other than | |
| **position: static**   #t2 {        background-color: darkmagenta;        color: white;        height: 500px;        top: 400px;        position: static;      } | Static means, same position, NO properties (top,left) will be applicable  As by default it is static even if u give position: static; or not, there will not be any change  Note: here top property is not at all applicable here, its waste of giving  Even if u give also top: 400px; will not come |
| **position: absolute**   <style>      #navbar {        background-color: coral;        top: 90px;        left: 50px;        position: absolute;        /\* Here for this tag parent is body Hence this tag        moved 90px from body top & 50px from left body  \*/      }      #offer {        background-color: lightskyblue;        height: 100px;      }      body {        background-color: antiquewhite;      }    </style>    <body>      <h2 id="navbar">        <span>Myntra </span>        <span>Men </span>        <span>Women </span>        <span>Kids </span>        <span>Home and Living </span>        <span>Beauty </span>        <span>studio </span>      </h2>    </body> | This means these properties(top,left) will be applied based on absolute(parent tag position)  if no parent found then, these properties will be applied from the body tag    For the <h2 id="navbar"> we gave top as 90px from absolute position (means from parent where parent is the whole body tag) |
| **position: relative**        #t2 {        background-color: darkmagenta;        color: white;        height: 50px;        top: 150px;        left: 50px;        position: relative;        /\* Here relative means the properties (top)        will be applied from actual position \*/      }   <h3 id="t2">2.See all the images</h3> | This means those properties (top,left) will be applied from actual position (undavalasina position)  Always actual position can be derived by keeping static    When we keep static that element is at original position  When we keep as relative , all the properties are applied from the original position  In position: absolute the properties will be applied from parent tag |
| **position: fixed**   #navbar {        background-color: coral;        position: fixed;        /\* Here for this tag parent is body Hence this tag        moved 90px from body top & 50px from left body  \*/      }    <h2 id="navbar">        <span>Myntra </span>        <span>Men </span>        <span>Women </span>        <span>Kids </span>        <span>Home and Living </span>        <span>Beauty </span>        <span>studio </span>      </h2> | Fixed means that element will be in fixed state- it wont move even if you scroll  Note:- when it is in fixed state, the other ele will assume that ele is not present at all |
| position: fixed;  #t2 {        background-color: darkmagenta;        color: white;        top: 200px;        height: 500px;        left: 50px;        position: sticky;      } | Means it will become sticky/fixed only when certain condition is met  Here when for that element if top is 200px then that element will be sticky |

Css units

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Px(pixels)  Related to absolute size | % (related to parent width/height) | vw/vh  Related to screen size | Rem(root element font)  Rem by default it will be 16 pixels  Here root element is html  Html{  Font-size: 16px  } | Em  (related to current element font size) |

Note:- default width will be screen width

|  |  |
| --- | --- |
| Percentage :- This will always be calculated based upon parent tag   <style>      #parent {        background-color: antiquewhite;        height: 300px;      }      #i1 {        background-color: aqua;        width: 250px;        height: 30%;  /\* Here i1 will be 30% of parent 300px \*/      }      #i2 {        background-color: sienna;        height: 60%;  /\* Here i2 will be 60% of parent 300px \*/      }    </style>    <body>      <h2 id="parent">        <div id="i1">TCS is safe workplace</div>        <div id="i2">Infosys too</div>      </h2>    </body> | Here i1 will be around 30%, i2 will be around 60%  So remaining 10% Is still in biscuit colour |
| VW (viewing width)  , VH (view height)  <style>      #parent {        background-color: antiquewhite;        height: 300px;      }      #i1 {        background-color: aqua;        height: 10vw;        /\* Here i1 height will vary depends on screen width\*/      }      #i2 {        background-color: sienna;        height: 60vw;        /\* Here i2 height will vary depends on screen width\*/      }    </style>    <body>      <h2 id="parent">        <div id="i1">TCS is safe workplace</div>        <div id="i2">Infosys too</div>      </h2>    </body> | These values will vary based on screen size   1. In myntra website all images size both width & height depends on viewing height and width, if tab area /screen size is reduced, image area is also getting small |
| <style>      #parent {        background-color: antiquewhite;        height: 800px;        width: 400px;      }      #i1 {        background-color: aqua;        height: 10vh;        /\* Here i1 height will be 10% of screen height\*/        width: 90vw;        /\* Here i1 width will be 90% of screen width \*/      }      #i2 {        background-color: sienna;        height: 60vh;        /\* Here i2 height will 60% of screen height\*/        width: 10vw;        /\* Here i2 width will 10% of screen width\*/      }    </style>    <body>      <h2 id="parent">        <div id="i1">TCS is safe workplace</div>        <div id="i2">Infosys too</div>      </h2>    </body> |  |
| Rem (root element font)  Here the default root element font is 16px, u can change    html {        font-size: 20px;        /\* This is the root element font  \*/      }     #rootEleFontDemo {        width: 20rem;        /\* Here this width will vary depending upon root element font \*/        /\* 20rem=20\*20=400px will the width \*/        /\* width: 400px; \*/        background-color: peru;      }      #px400 {        background-color: darkorchid;        width: 400px;      } | We change width & height based on root element font  If font increases like width also will be increased  Here root element is html  <div id="rootEleFontDemo">        This div width varies on root element font =20rem =20\*20px=400px width      </div>      <div id="px400">This div is 400px</div>    </body>    If u observe both these are having same width  Instead of giving  Note:- when html root font size increases then that element width also will be increased  When rem is 30 |
| When root element font is increased, the width also will be increased automatically   html {        font-size: 25px; /\* Now changed to 25px\*/        /\* This is the root element font  \*/      }      #rootEleFontDemo {        width: 20rem;        /\* 20rem=20\*20=400px will the width \*/        /\* width: 400px; \*/        background-color: peru;      } | Now as root element font increased the **width also increased automatically** |
| em  (Based on current font)  #currentEleFontDemo {        background-color: palegreen;        font-size: 20px;        width: 20em;        /\* Here this width will vary depending upon current ele font \*/      }   <div id="currentEleFontDemo">        This div width varies on root element font =20rem =20\*20px=400px width      </div> | When current ele font increases the width also will be increased   #currentEleFontDemo {        background-color: palegreen;        font-size: 30px;        width: 20em;        /\* Here this width will vary depending upon current ele font \*/      } |

Transitions

In real life, like KT transition means a phase where knowledge will be transferred, it will not happen immediately it will take some time,

Within that much time transition will happen

When u apply a transition, instead of applying effect immediately, effect will be applied slowly in graphics

Imp Note:- while applying some transitions, that div it should have some initial amount

Ex:- existing div should have height 10px, and on hover u can change to 200px.. , but existing if height is not mentioned transition will not be applied

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | #heading {  **height: 10vh;**          border: 3px dashed;          background-color: aqua;  **transition: all 4s;**        } | #heading:hover {  **height: 12vh;**          background: linear-gradient(            to right,            rgb(20, 141, 240),            rgb(248, 244, 30),            rgb(12, 226, 101)          );        } |   Here initially height is 10vh, on hover it will increase to 12vh, whereas it will not apply immediately, it will take 4s time to apply  // Here all means, all the changes will take 4s to apply, like height, width changes  **transition: <any property name here> <time> <transition-timing-fun> <delay> ,**  **<any property name here> <time> <transition-timing-fun> <delay>;**  **Ex:- transition: all 4s;**  **transition: width 4s, height 4s;**  **like height,width** | <div id="heading">        <div id="main">Prakash Inverter Solution's</div>        <div id="caption">          Any Inverter Any problem One solution ~~It's Prakash inverter        </div>      </div>    From this to below it will take 4s that is the transition |
| **transition-timing-function**   |  |  | | --- | --- | | Linear | Linear means all the effect will be applied slowly  This is the default one | | Ease | Consider ease as slow | | Ease-in | Starting slow | | Ease-out | Ending slow | | Ease-in-out | Slow-fast-slow | |  |
| **Transition delay**  **Instead of applying transition immediately,**  **Transition will be applied after some delay** |  |

Issue

|  |  |
| --- | --- |
| #seven {        background-color: aqua;        transition: all 4s;        height: 50px;      }      #seven:hover {        height: 100px;        /\* here height is applying slowly  but gradient col is applying immediately \*/        background: linear-gradient(          to right,          rgb(89, 235, 76),          rgb(238, 25, 238),          rgb(245, 178, 77)        );      }      <div id="seven">Mani</div> | Initial    Prblm is gradient applying immediately |
|  |  |

Transformations

|  |  |
| --- | --- |
| Translate- Moves an element sideways or up and down. | Scale – increase height and width |
| Rotate | skew |

When we apply this transformations, the elements will be transformed/changed

* **translate():** Moves an element sideways or up and down.

**Note:**- even for this transformations, u apply transitions so that slowly these will change (instead of immediately applying the transformations)

Always negative values means left/top (ex:- x-axis left, y-axis top)

|  |  |
| --- | --- |
| Moving down and right (both x,y are positive values)   #one {          background-color: violet;          transition: all 4s;          /\* With this transition all           transformations will be applied slowly \*/        }        #one:hover {          transform: translate(30px, 100px);          /\* here x=30px right ,y=100px down \*/        } | Here after applying transformation ele went 30px right and 50px down from current position |
| Moving left and top bec of negative values   #one {          background-color: violet;          transition: all 4s;          /\* Bec of this transformation will happen in 4s \*/          padding: 50px 50px;          border: 2px dashed;          margin: 100px;          box-sizing: border-box;          /\* With this transition all           transformations will be applied slowly \*/        }        #one:hover {          transform: translate(-30px, -100px);          /\* here x=-30px means left ,y=-100px means top \*/        } |  |
| Giving in %  transform: translate(-50%, -100%);  now that div will be moved to left and top |  |
| Moving only on x-axis towards right for % values  #one:hover {          transform: translateX(80%);        }  transform: translateX(200px);  transform: translateX(-2000px);  it will go 2000px left but if it is already at left it will go but u cant see | Moving in yaxis – top 2000px  transform: translateY(-2000px); |

* Scale- to increase the size like height and width of an element

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Scale  Increase both width height  Always 1st x axis and then y axis | |  |  |  | | --- | --- | --- | | #two {          background-color: coral;          margin: 50px;        } | #two:hover {          transform: scale(2);        }  Means on hover current ele will be increased 2 times of its current height,width | #two:hover {          transform: scale(2, 3);        }  Means on hover current ele will be increased  2 times horizontally  3 times vertically | |
| scaleX()  It will grow horizontally 2 times | #two:hover {          /\* transform: scale(2); \*/          transform: scaleX(2);        } |
| scaleY()  it will grow vertically 2 time  horizontally it will be same | #two:hover {          /\* transform: scale(2); \*/          /\* transform: scaleX(2); \*/          transform: scaleY(2);        } |

* Rotate

|  |  |
| --- | --- |
| Rotate (45 deg) –both x,y axis will be rotate 45deg  Rotate(45 deg,45 deg)  Rotate in both x-axis,y-axis | #three {          background-color: greenyellow;          margin: 50px;        }        #three:hover {          transform: rotate(45deg);          transform: rotateX(45deg);          transform: rotatey(85deg);        } |
| transform: rotateX(45deg); here height will be compressed  (just move up,down) |  |
| transform: rotateY(45deg) here Y-axis means vertical, Y-axis it will remain constant  here width will be compressed, | here width is compressed |
|  |  |

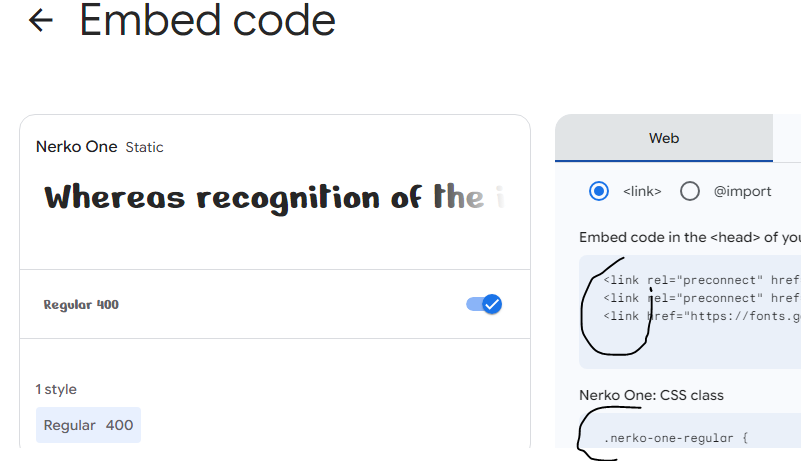
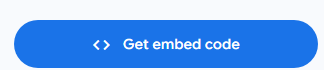
Skew – means slant

|  |  |  |  |
| --- | --- | --- | --- |
| By default – skew means x axis movement  transform: skewX(24deg); == transform: skew(24deg);  +ve means it went horizontally right  skewX- side wards movement  #four {          margin: 100px;          background-color: rgb(224, 59, 100);          transition: all 4s;        }        #four:hover {          transform: skew(24deg);          /\* it went horizantally right side \*/        } | |  |  | | --- | --- | | #four:hover {          transform: skew(-24deg);          /\* as negative it went horizantally left side \*/        } |  | |
| skewY(44deg) – a vertical movement bec of +ve value it went top   |  |  | | --- | --- | | #four {          transform: skewY(44deg);          /\* as +ve movement in y axis to vertically up\*/          margin: 100px;          background-color: rgb(224, 59, 100);        } |  | | |  |  | | --- | --- | | #four:hover {          transform: skewY(-24deg);   /\* as negative it went vertically down side \*/        } |  | |
| Both vertical and horizontal movement  #four {          transform: skew(24deg, 24deg);          /\* a movement in y axis up and down\*/          margin: 100px;          background-color: rgb(224, 59, 100);        } | Both horizontally left, 45 deg top |

Google fonts

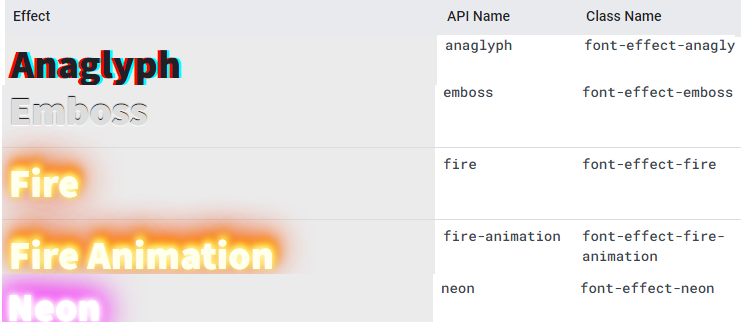
|  |  |
| --- | --- |
| API docs for developers – it has all fonts will all effects | <https://developers.google.com/fonts> |

<https://fonts.google.com/specimen/Nerko+One>

Goto any font on top right side click “Get font” & click 

|  |
| --- |
| U should add the links and copy the style from above |
| <head>      <meta charset="UTF-8" />      <meta name="viewport" content="width=device-width, initial-scale=1.0" />      <title>Google font</title>  **<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=Fira+Sans:ital,wght@0,100;0,200;0,300;0,400;0,500;0,600;0,700;0,800;0,900;1,100;1,200;1,300;1,400;1,500;1,600;1,700;1,800;1,900&display=swap"**  **rel="stylesheet"**  **/>**      <style>  **.nerko-one-regular {**  **font-family: "Nerko One", cursive;**  **font-weight: 400;**  **font-size: 80px;**  **font-style: normal;**  **}**      </style>    </head> |
| <body>      <div class="nerko-one-regular">Bro</div>    </body> |

Font effect



If u want the fire effect just

To use this beta feature, simply add effect= to your Google Fonts API URL and add the corresponding class name to the HTML element(s) that you want to affect. In our example above, we used the shadow-multiple font effect, so the request would look like:

1. Prepare the url (https://fonts.googleapis.com/css?family=Rancho&effect=fire-animation) with effect=API name (API name is already provided in <https://developers.google.com/fonts/docs/getting_started> )

Ex: - For example if u want fire animation, prepare the url as

https://fonts.googleapis.com/css?family=Rancho&**effect=<API-Name>**

https://fonts.googleapis.com/css?family=Rancho&effect=fire-animation

1. And add provided class to any div (the effect u can modify )

Ex:-

<div class="**font-effect-fire-animation**" id="one">Bro</div>

Here if font size is low, u can write one more class and add that class beside it

|  |  |
| --- | --- |
| <head>   <link        rel="stylesheet"        href="https://fonts.googleapis.com/css?family=Rancho&effect=**fire-animation**"      />  </head>  #one {          font-size: 80px;        }   <div class="**font-effect-fire-animation**" id="one">Bro</div> |  |
| Here in url effect=APIName as per above table in <https://developers.google.com/fonts/docs/getting_started>  And class name is already mentioned in table, |  |
| <link        rel="stylesheet"        href="https://fonts.googleapis.com/css?family=Rancho&effect=fire"      />   .big {          font-size: 40px;        }   <div class="font-effect-fire big">Service Engineer -Prakash</div> |  |
| If u are using multiple effects use pipeline symbol and add in same url instead of adding multiple urls    <link        rel="stylesheet"        href="https://fonts.googleapis.com/css?family=Rancho&effect=**neon | fire | 3d-float"**      /> |  |

Box model

There are 2 types

|  |  |
| --- | --- |
| Content box model (width will be only content)  (The default is content box  ) | When u choose the box model as content-box the width& height will be only for  Content  This means only the width, height will only be applicable for content – not for padding, border and all  So if u add padding, border width and all the total width will be increased again |
| Border box model (width will be including border)   <style>      html {        font-size: 20px;      }      #bb {        background-color: yellowgreen;        font-size: 2rem;        /\* by default 1 rem root ele font = 16px        so here final font size will be taken as  =32 px \*/        width: 400px;        height: 100px;        /\* as current font size =32px then width here is 320px \*/  **box-sizing: border-box;**        padding: 10px;        border: 15px dashed darkgoldenrod;      }      #cb {        box-sizing: content-box;        font-size: 2rem;        background-color: salmon;        width: 400px;        padding: 10px;        border: 20px dashed royalblue;        height: 100px;      }      #px320 {        background-color: pink;        width: 400px;      }    </style>    <body>      <div id="bb">Charan is lazy fellow</div>      <div id="px320">Santhoshi is lazy fellow 400px</div>      <div id="cb">Prakash is hard worker</div>    </body> | Here the values which we will give for width and height will be for all {content+padding+border}    When u choose the box model as border-box the width& height will be including  Content+border+padding,   1. Here in above example- Prakash content comes under content box model,   Here width will only be content  so when we gave padding, margin – the width went beyond 400px – because as this is content box only content will be 400px   1. Charan comes under border box- means – width =400 means , content +padding+border everything will fit under the same width |

Box shadow

Box-shadow:

|  |  |
| --- | --- |
|  | <style>      #one {        background-color: rgb(209, 192, 40);  <horizontally left (- values)/right (positive values)> <vertically top(negative value)/bottom (positive values)> <blur value> <spread distance> <color>        margin: 300px;  **box-shadow: 10px -10px 10px 10px grey;**        height: 200px;        width: 200px;        padding: 60px;  **box-sizing: border-box;**        font-family: sans-serif;      }    </style>    <body>      <div id="one">Charan</div>    </body> |

Colors

We can give colors using 6 ways

|  |  |
| --- | --- |
| Direct color name | background-color: salmon; |
| RGB function- red green blue  U will get all colours In universe by providing these 3 values | rgb(200, 150, 21); |
| Using hexa decimal number  Where 1st 2 letters represents Red,  next 2 numbers represents Green,  last 2 letter indicates Blue | background-color: #23Ab95;  this will give us some dark green colour |
| HSL (hue, saturation, lightness) | Hsl(10%,20%,50%) |
| RGBA here last a represents opacity(opposite of transparency) | Rgba(255,0,1,0.6) |
| HSLA | Hsla(10,20%,50%,0.6) |

Gradients backgrounds

Linear gradient, radial-gradient

|  |  |
| --- | --- |
| linear-gradient | radial-gradient |
| repeating-linear-gradient | repeating-radial-gradient |

U can apply these gradients only to a color or an image

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | //Here the colors will move from right to left   .two {        background: linear-gradient(  **to left,**          rgb(169, 228, 53),          rgb(59, 191, 231),          rgb(228, 53, 135)        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      } | // from bottom to top   .two {        background: linear-gradient(  **to top,**          rgb(169, 228, 53),          rgb(59, 191, 231),          rgb(228, 53, 135)        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      } | //u can even apply diagonally  --diagonally left to right   .two {        background: linear-gradient(  **to bottom right,**          rgb(169, 228, 53),          rgb(59, 191, 231),          rgb(228, 53, 135)        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      } | //diagonally left bottom right to top left   .two {        background: linear-gradient(          to top left,          rgb(169, 228, 53),          rgb(59, 191, 231),          rgb(228, 53, 135)        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      } |
| U can even provide degrees  .two {        background: linear-gradient(          -45deg,          rgb(169, 228, 53),          rgb(59, 191, 231),          rgb(228, 53, 135)        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      } | Radially gradient- means circularly   .one {        background: radial-gradient(          rgb(169, 228, 53),          rgb(59, 191, 231),          rgb(228, 53, 135)        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      }    <div class="one">Manideep</div> | //Repeating radial  In repeating u have to mention %  .three {        background: repeating-radial-gradient(          rgb(190, 231, 41) 10%,          rgb(225, 59, 231) 20%,          #2f5566 20%        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      }    <div class="three">Charan</div> | //Repeating linear gradient  .four {        background: repeating-linear-gradient(          45deg,          rgb(169, 228, 53) 10%,          rgb(59, 191, 231) 20%,          rgb(228, 53, 135) 20%        );        height: 200px;        width: 200px;        padding: 80px 60px;        box-sizing: border-box;      }  <div class="four">SAI prakasham</div> |  |
|  |  |  |  |  |

Gradients Text

To make text as gradient, we have to give 3 properties (background, color,       -webkit-background-clip)

If any property is missed nothing will come in output

|  |  |
| --- | --- |
| .five {        height: 200px;        width: 900px;        font-weight: bolder;        font-size: 80px;        background: linear-gradient(          to right,          rgb(76, 203, 235),          rgb(235, 65, 198),          rgb(77, 245, 166),          rgb(245, 178, 77)        );        color: transparent;        -webkit-background-clip: text;      }    </style>      <div class="five">Charan Ga chaduvura</div> |  |

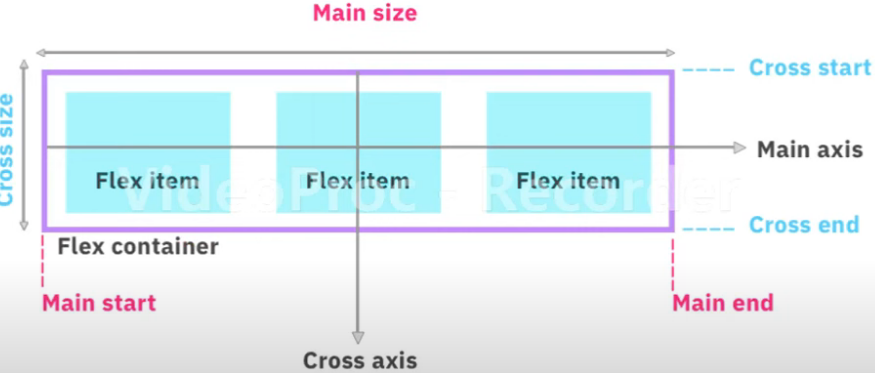
Z-Index

Flex boxes

This flexbox = table++, outer box is called parent container, and all remaining ele are called flex items

|  |  |  |  |
| --- | --- | --- | --- |
| This flex box have more features than a table  EX:- in table we can place data only row-wise, here we can place  row-wise,  column-wise,  We can reverse a row data very easily using Flex-direction: row-reverse  Sample code   |  |  | | --- | --- | | <style>        .parent {          display: flex;          flex-direction: row-reverse;          border: 5px dashed salmon;          flex-wrap: wrap;        }        .child {          width: 200px;          height: 100px;        }        .c1 {          background-color: aqua;        }        .c2 {          background-color: chartreuse;        }        .c3 {          background-color: coral;        }      </style> | <body>      <div class="parent">        <div class="c1 child">one</div>        <div class="c2 child">two</div>        <div class="c3 child">three</div>        <div class="c1 child">four</div>        <div class="c2 child">Five</div>        <div class="c3 child">six</div>        <div class="c1 child">seven</div>        <div class="c2 child">Eight</div>        <div class="c3 child">Nine</div>      </div> | |  |

Parent properties



|  |  |
| --- | --- |
| display:flex |  |
| flex-direction: row | Main axis will be horizontal (x-axis) |
| flex-direction: column | Then main axis will be vertical / Y-axis |
| justify-content: center;  align-items: center; | To move any item horizontally & vertically center |
|  |  |

|  |  |
| --- | --- |
| All these below properties are parent properties | |
| display:flex |  |
| Flex-direction: row, row-reverse, column, column-reverse | For a flex box, default direction is row |
| flex-wrap: nowrap | wrap | wrap-reverse  If u wrap the child properties like height, width will be retained, .  Else elements will be crushed    Without wrapping all elements will be crushed & adjusted in same row | Same like wrap, if all flex items it doesn’t fit into single row , the elements will be moved to next row or column by maintaining mentioned ele width, height  If wrap is not applied all flex items will be compressed to place in a single row  Wrap-reverse- means bottom  With flex-wrap elements will be moved to next row |
| flex-flow: <'flex-direction'> || <'flex-wrap'> | This is just a shorthand operator- of direction,wrap |
| justify-content: flex-start | flex-end | center | space-between | space-around| space-evenly  (this will talk about the spacing/placement in main axis)  If flex-direction: row; then main axis is horizontal  .parent {          display: flex;          flex-direction: row;          border: 5px dashed salmon;          flex-wrap: wrap;          justify-content: flex-end;          /\*This is same like right aligned  \*/        } | This is based on main axis  When u want a space between each element then prefer below   |  |  | | --- | --- | | Space-around | Means Same space will be given to the left side and right side of each element | | Space-between | Means total space will be only in between those elements  Not left side and right side | | Space-evenly | Means between each element same space will be given  Total available space will be divided and split evenly across all elements | | Flex-start | Means elements will be starting from left | | Center | Means remaining elements will be moved to center | | Flex-end | Means remaining elements will be moved to right side |     Means same space between left and right side |
| align-items: flex-start | flex-end | center | baseline | stretch;  ex:- | When we gave flex-start, when flex-direction: row, the 2nd line elements will come to left like below |

**Align-content**:

|  |  |
| --- | --- |
| align-content:  flex-start | flex-end | center | space-between | space-around | stretch  (this will talk about the spacing in cross/opposite axis)  If flex-direction: row; then cross-axis will be y-axis as Main axis will be x-axis |  |

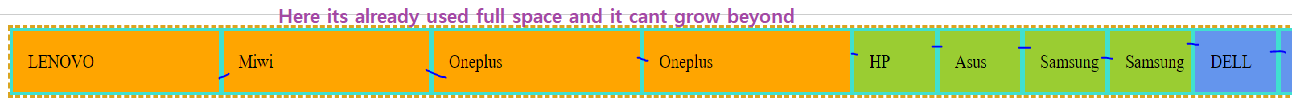
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| .parent {        display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;      } | .parent {       display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;  **align-content: flex-start;**  /\* Bec of this elements are vertically moved to top/start \*/      } | .parent {        display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;        align-content: center;        /\* Bec of this elements are vertically centered \*/      } | .parent {        display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;        align-content: flex-end;        /\* Bec of this elements are vertically moved to end \*/      } |  |
|  |  |  |  |  |
| .parent {        display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;  **align-content: space-around;**        /\* Bec of this vertical space is from around each elment \*/      } | .parent {  flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;  **align-content: space-between**;        /\* Bec of this vertical space        is from between each elment not before        and after end\*/      } | .parent {        display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;        align-content: space-evenly;        /\* Bec of this vertical space        is from between each elment        is even/same\*/      } |  |  |

Child properties

|  |  |  |
| --- | --- | --- |
| Order: integer  (This is like rank how students get)   |  | | --- | | <style>        .parent {          border: 3px dashed goldenrod;          display: flex;          flex-wrap: wrap;        }        .child {          background-color: burlywood;          border: 3px solid turquoise;          margin: 20px;          padding: 15px;          width: 50px;        }        .c1 {          background-color: cornflowerblue;          order: 1;        }        .c2 {          background-color: yellowgreen;          order: 3;        }        .c3 {          background-color: orange;          order: 2;        } | | U can give order to each element,  so that low order element will be placed 1st, higher order rank will be moved to last    Here we gave order 1 for all blue color so all blue color boxes came first   <body>      <div class="parent">        <div class="c1 child">DELL</div>        <div class="c2 child">HP</div>        <div class="c3 child">LENOVO</div>        <div class="c1 child">Logitech</div>        <div class="c2 child">Asus</div>        <div class="c3 child">Miwi</div>        <div class="c1 child">Redmi</div>        <div class="c2 child">Samsung</div>        <div class="c3 child">Oneplus</div>      </div>    </body> |

|  |  |
| --- | --- |
| Flex-grow: number   .c3 {          background-color: orange;          order: 2;          flex-grow: 2;        }  <div class="c3 child">Oneplus</div>  <div class="c3 child">LENOVO</div>  <div class="c3 child">Miwi</div>  As we applied  flex-grow: 2; those ele took space and those elements grown 2 times than other elements | This indicate that element will be grown that many units size so total available size will be divided into units and based on the number that div will be expanded |

Note:- once it used full space it cant& wont grow beyond even if u give    flex-grow: 2000;Here already there is no space left between those 2 elements, hence 2000 also doesn’t make any huge difference



|  |  |
| --- | --- |
| Flex-shrink: number  .c1 {  background-color: cornflowerblue;  order: -1;  /\* Bec of order blue ele came 1st \*/  flex-shrink: 30;  /\* Bec of shrink these ele shrinked \*/  } | When flex-wrap: nowrap, then all elements will be placed in same row & if size is not fitted then all ele will be shrinked,  if u give flex-shrink:10 , then that particular element will be shrinked 10 times than other elements  Here if u see only blue color elements shrinked more |
| Flex-basis: length | auto |  |
| flex: ‘flex-grow’, ‘flex-shrink’, ‘flex-basis’ |  |
|  |  |

#### Align-self

justify-content talks about main-axis (if flex-direction: row then main axis will be x-axis)

Align-content/ align-self talks about cross-axis (if flex-direction: row then main axis will be y-axis)

As we are applying this property to child individual element (not to parent container) –that individual element cross axis position is changed

|  |  |
| --- | --- |
| align-self: auto,flex-start,flex-end,center,baseline,stretch  (this align always talks about vertical /cross axis)   <style>        .parent {          border: solid 3px slateblue;          display: flex;          justify-content: center;          width: 95vw;          height: 95vh;          font-family: "Trebuchet MS", "Lucida Sans Unicode", "Lucida Grande",            "Lucida Sans", Arial, sans-serif;          font-size: 80px;          background-color: moccasin;          align-content: center;        }        .child {          align-self: center;        }      </style>  <body>      <div class="parent">        <div class="child">Charan</div>      </div>    </body> | We can exactly place at center |

|  |  |  |  |
| --- | --- | --- | --- |
| These properties are applied to parent | | | |
| .parent {          display: flex;          justify-content: flex-end;  /\* Bec of flex-end hori ele were moved to end \*/  } |  | .parent {          display: flex;          justify-content: center;          /\* Bec of flex-end hori ele were moved to center \*/ |  |
|  | | | |

What is the diff between   child’s   align-self: center; && parent’s align-content: center;

When u can this property to a particular child – if u have many child elements/flex-items, u can bring that particular child to down

The only difference is with child’s   align-self: center u can apply that vertical alignment to that particular child

|  |  |
| --- | --- |
|  | With   #three {          align-self: center;        }  <div id="three" class="child">Bing</div>  With a u can bring that particular child to down  Whereas if u apply that to parent /flex-container then changes will be applied to all the elements  Here as we applied to particular child called “Bing” that particular 3rd element alone came to center |
| <style>        .parent {          display: flex;          justify-content: center;          flex-wrap: wrap;          width: 75vw;          height: 75vh;          font-family: "Trebuchet MS", "Lucida Sans Unicode", "Lucida Grande",            "Lucida Sans", Arial, sans-serif;          font-size: 7vw;          background-color: rgb(187, 236, 50);          border: solid 3px rgb(138, 127, 209);        }        .child {          border: 4px dashed;          margin: 2px;          /\*Bec of flex-start this div is moved to vertical middle \*/        }        #parent > div:hover {          background-color: coral;          align-self: center;  /\*here bec of this when ever a child div is hovered that particular element will be moved to center , here for ex I hovered on 1st ele “c” so it went down \*/        }      </style> | |  |  | | --- | --- | |  | <body>      <div id="parent" class="parent">        <div id="one" class="child">C</div>        <div id="two" class="child">h</div>        <div id="three" class="child">a</div>        <div id="four" class="child">r</div>        <div id="five" class="child">a</div>        <div id="six" class="child">n</div>      </div>    </body> | |

|  |  |  |  |
| --- | --- | --- | --- |
| These properties are applied to child | | | |
| .child {          border: 4px dashed;          margin: 2px;          align-self: flex-start;          /\*Bec of flex-start this div is moved to vertical top \*/        } |  | .child {          border: 4px dashed;          margin: 2px;          align-self: flex-end;          /\*Bec of flex-start this div is moved to vertical BOTTOM \*/        } |  |
| .child {          border: 4px dashed;          margin: 2px;          align-self: center;          /\*Bec of align-self this div is moved to vertical middle \*/        } |  | | |
| The above can also be done using    <style>        .parent {          display: flex;          justify-content: center;  /\*bec of this horizontal middle\*/          flex-wrap: wrap;          align-content: center;        .child {          border: 4px dashed;          margin: 2px;          /\*Bec of flex-start this div is moved to vertical middle \*/        } | I have bought this text “Google” to center using 2 ways   1. align-content: center; (parent property) 2. align-self: center; (child’s property) | | |

Vertically center

|  |  |
| --- | --- |
| Bringing vertically centre using parent properties | Using child properties |
| u need to use both  flex-wrap: wrap; && align-content: center; (I don’t know why wrapping is mandatory to bring ele to center) | u have to use align-self: center; for each child |
| #parent {  font-size: 40px;  color: violet;  display: flex;  /\* flex-direction: row; \*/  flex-wrap: wrap;  justify-content :center;  align-content: center;  border: 3px dashed seagreen;  height: 90vh;  }  #child{  /\* align-self: center; \*/  border: 3px dashed turquoise;  }  body{  border: 3px dashed tomato;  background-color: wheat;  } | #parent {  font-size: 40px;  color: violet;  display: flex;  flex-direction: row;  justify-content :center;  flex-wrap: wrap;    border: 3px dashed seagreen;  height: 90vh;  }  #child{  align-self: center;  border: 3px dashed turquoise;  }  body{  border: 3px dashed tomato;  background-color: wheat;  } |

Vertically& horizon center

Any items if u want to move vertically and horizontally center then

1. make it as a flex box by adding display: flex;
2. apply these properties

justify-content: center;

align-items: center;

|  |  |  |
| --- | --- | --- |
|  | Here we selected all items/div’s in that parent tag and making each ele as a flexbox  #parent > div {          display: flex;          justify-content: center;          align-items: center;        } | <div id="parent">        <div id="box">C</div>        <div id="box">H</div>        <div id="box">A</div>        <div id="box">R</div>        <div id="box">A</div>        <div id="box">N</div>      </div> |

Media queries

Depending upon screen width we changed the background color

*/\* This says if the screen width is 500-800px then this color will be applied \*/*

@media screen and (min-width:500px) and (max-width:800px) {

    body{

        background-color: greenyellow;

    }

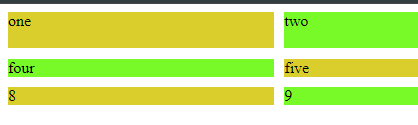
}

Grid

For grid we use names as items – ex:- justify-items, align-items

Main advantage of grid over flex box is 1) column gap and row gap 2) a cell can extend/exists across many rows/columns

And same like flex boxes- u can horizontally move /vertically move the grid left/right/up/down



Note:- while coding always try to give background for grid, so that u will understand till where the grid resides

Parent properties

|  |  |
| --- | --- |
| display: grid |  |
| Grid-template-columns  Grid-template-rows | Here u can mention each col width and each row height |
|  |
| Grid-template |  |
| Grid-template-areas | Here u can mention the layout |
| Grid-row-gap  Grid-column-gap | Gap between the rows and columns |
|  |
| Grid-gap |  |
| Justify-items | This means ele will be horizontally compressed  Default is stretch |
| Align-items | This means ele/items will be vertically compressed |
| Justify-content | Grid/Content will be moved horizontally |
| Align-content | Grid/Content will be moved vertically |
| Note:- u can do any layout using 2 ways   1. using grid-template-areas (parent property) & gird-area (child property) or 2. grid-row & grid-column (parent properties) – This is the best and easy way -bec here we don’t need to write un-necessarily meaningless properties (grid-area) | |

U can give some background-color/border to parent grid , so that u will understand till where the border is

|  |  |
| --- | --- |
| display: grid |  |
| grid-template-columns | */\* grid-template-columns: 100px 150px 250px; \*/*              grid-template-columns: 1fr 3fr 1fr;  u can give value as “auto” also  //here we should mention the width of each column either in px or in fractions (fr)  If u don’t mention data will not appear in a grid it will normally appear as div’s  With this we will mention the number of columns & each column width  Here as we have 3 columns :- if we sum totally 1+3+1 = 5fr totally 5 parts /100  grid-template-columns: 100px 100px auto;  in this case 3 column will occupy full width instead of 100px auto means it will take what ever it is available |
| grid-template-rows | With this we will mention the number of rows & each row height  grid-template-rows: 2fr 1fr 1fr;  grid-template-rows: 2fr 1fr auto;  we can give auto also |
| grid-template |  |
| grid-template-areas | |  |  | | --- | --- | |  |  | |
| grid-row-gap | This is the gap between 2 rows  row-gap: 1cap; |
| grid-column-gap | This is the gap between 2 columns  column-gap: 1cap; |
| grid-gap/ gap | To mention gap between each row & column  gap: 1cap; bec of this gap will come inside grid    This is the gap that will come between each column & start |
| justify-items  The default value is stretch | This says elements will be horizontal compressed  justify-items: center; //here ele will be compressed towards center  means by default all elements will be compressed towards center     |  |  |  | | --- | --- | --- | | justify-items: center; | justify-items: end;  here compressed towards right/end | justify-items: stretch;  no compression | | justify-items: center;  horizontal compression towards center | justify-items: start;  Horizontally compression towards start/left | justify-items: end;  horizontally compressed towards right/end | |
| align-items  By default the value is stretch | This says items/elements will be vertically compressed  Start, stretch, center, end   |  |  | | --- | --- | |  | align-content: start; //- ele are vertically centered  align-items: center; // means ele are vertically compressed | |
| Align-content:  Default is start | To vertically move that grid up/down  */\* align-content: start;*  *align-content: center; \*/*              align-content: end; |
| Justify-content | To horizontally move that grid left/right  justify-content: start;              justify-content: center;              justify-content: end;   |  |  |  | | --- | --- | --- | | justify-content: center;  entire grid moved to center | justify-content: start;  Entire Grid moved to left | Entire grid moved to right  justify-content: right; | |  |  |  |  |  |  | | --- | --- | | align-content: center;  justify-content: center; |  | |

Align-items :- This talks about vertical movement

|  |  |
| --- | --- |
| <style>          #parent {              display: grid;  */\* grid-template-columns: 100px 150px 250px; \*/*              grid-template-columns: 0.5fr 3fr 1fr;              grid-template-rows: 5fr 1fr 1fr;              column-gap: 1cap;              row-gap: 1cap;              border: 3px dashed **rgb**(152, 24, 216);              align-items: stretch;          }          #parent>div:nth-child(even) {              background-color: **rgb**(120, 250, 40);          }          #parent>div:nth-child(odd) {              background-color: **rgb**(218, 206, 44);          }          .child {              border: 3px dashed palevioletred;          }      </style> | <div *id*="parent">          <div *id*="one" *class*="child">one</div>          <div *id*="two" *class*="child">two</div>          <div *id*="three" *class*="child">three</div>          <div *id*="four" *class*="child">four</div>          <div *id*="four" *class*="child">five</div>          <div *id*="four" *class*="child">seven</div>          <div *id*="four" *class*="child">8</div>          <div *id*="four" *class*="child">9</div>          <div *id*="four" *class*="child">10</div>      </div> |

|  |  |  |  |
| --- | --- | --- | --- |
| Default is stretch | align-items: start;  this means vertically top/start | align-items: center;  vertically center | align-items: end;  vertically bottom |
|  |  |  |  |

Justify-items: This is about horizontal movement

|  |  |  |  |
| --- | --- | --- | --- |
| justify-items: stretch;  default is stretch – it will occupy full width | justify-items: start;  horizontally left | justify-items: center;  horizontally center | justify-items: end;  horizontally right |
|  |  |  |  |

While practicing, make sure u use larger height, width

 #parent {

            display: grid;

            width: 90vw;

            height: 30vh;

*/\* grid-template-columns: 100px 150px 250px; \*/*

            grid-template-columns: 0.5fr 3fr 1fr;

            grid-template-rows: 10fr 3fr 3fr;

            column-gap: 1cap;

            row-gap: 1cap;

            border: 3px dashed **rgb**(152, 24, 216);

*/\* align-items: end;*

*align-items: start;*

*align-items: end;*

*justify-items: end;*

*justify-items: start; \*/*

            justify-items: center;

*/\* This means horizontally center \*/*

            align-items: start;

*/\* This means vertically top \*/*

        }

|  |  |
| --- | --- |
|  | Top left 1st one is column-1 and  Top row is row-1  In this image we totally have 4 columns (4 vertical columns) and 4 rows (4 horizontal lines) |

Top

Child properties

|  |
| --- |
| Grid-column-start  Grid-column-end  Grid-column |
| Grid-row-start  Grid-row-end  Grid-row |
| Grid-area |
| Align-self |
| Justify-self |

|  |  |  |  |
| --- | --- | --- | --- |
| Grid-column-start  Grid-column-end  Grid-column | .one {              grid-column-start: 1;              grid-column-end: 1;          }   |  |  | | --- | --- | | Now the first image started and ended and 1st | grid-column-end: 3;  or  grid-column: 1/3; | |
|  |
| grid-column: 1/3;  this means this has started at column-1 and ended column-3 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Grid-row-start  Grid-row-end  Grid-row | |  |  |  | | --- | --- | --- | | .one {              grid-row-start: 1;              grid-row-end: 3;          }  Here this cell started at row -1 and ended at row-3 | .one {  */\* grid-row-start: 1;*  *grid-row-end: 3; \*/*              grid-row: 1/4;          }  Here this cell started at 1st row and ended at 4th row |  | |
|  | grid-row: 1/3;  this means this particular child started at 1st row and ended at row-3 |
| Grid-area |  |
| Align-self | align-self: self-end;  to move element vertically down |
| Justify-self | justify-self: center;  center,end,start  to horizontally compress that particular element   |  |  |  |  | | --- | --- | --- | --- | |  |  |  |  | |

Grid-template-areas

Main things here are grid-template-areas,             grid-area (for each and every class we have to apply grid-area property)

Note:- if u are using any class under grid-template-areas u have to include grid-area property for that class

            grid-template-areas: "one one one" //Here one,two.. these are all class names

                "two three four"

                "two three four"

                "five five five";

        .two {

            grid-area: two;

        }

            grid-template-areas: "one one ." if u give dot here empty will be there/ nothing will come

all the rows must be inside a quotation

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | <style>          #parent {              display: grid;  */\* Bec of this each column size will be 100px \*/*              grid-template-columns: 100px 100px 100px;  */\* Bec of this each row size will be 50px \*/*              grid-template-rows: 50px 50px 50px 50px;  */\* grid-template-areas: "child1 child1 child1", ""; \*/*              gap: 1cap;  */\* The below is like layout \*/*              grid-template-areas: "one one one"                  "two three four"                  "two three four"                  "five five five";          }          .child {              border: 3px dashed saddlebrown;          }          .two {              grid-area: two;          }          .three {              grid-area: three;          }          .four {              grid-area: four;          }          .five {              grid-area: five;          }          .one {              grid-area: one;          }          #parent>div:nth-child(even) {              background-color: greenyellow;          }          #parent>div:nth-child(odd) {              background-color: **rgb**(206, 46, 206);          }      </style> | <body>      <div *id*="parent">          <div *class*="one child">one</div>          <div *class*="two child">two</div>          <div *class*="three child">three</div>          <div *class*="four child">four</div>          <div *class*="five child">five</div>          <div *class*="six child">six</div>          <div *class*="seven child">seven</div>          <div *class*="eight child">eight</div>      </div>  </body> | |
|  |  |
|  |  |

Example code

|  |  |  |  |
| --- | --- | --- | --- |
|  | |  |  | | --- | --- | | <style>          #parent {              display: grid;  */\* Bec of this each column size will be 150px \*/*              grid-template-columns: 150px 150px 150px;  */\* Bec of this each row size will be 100px \*/*              grid-template-rows: 100px 100px 100px;              gap: 1cap;          }          .child {              border: 3px dashed saddlebrown;          }          .seven {              grid-column:1/4 ;  */\* Bec of this, 7th ele started at col-1, ended in col-4  \*/*          }          #parent>div:nth-child(even) {              background-color: greenyellow;          }          #parent>div:nth-child(odd) {              background-color: **rgb**(206, 46, 206);          }      </style> | <body>      <div *id*="parent">          <div *class*="one child">One</div>          <div *class*="two child">two</div>          <div *class*="three child">three</div>          <div *class*="four child">four</div>          <div *class*="five child">five</div>          <div *class*="six child">six</div>          <div *class*="seven child">seven</div>          <div *class*="eight child">eight</div>      </div>  </body> | |
|  |  |

#### Grid examples

|  |  |  |
| --- | --- | --- |
|  | |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | <style>        #parent {          display: grid;          background-color: rgba(226, 74, 99, 0.37);          grid-template-columns: 10vw 30vw 10vw;          grid-template-rows: 10vh 30vh 10vh;          justify-content: center;          gap: 0.2cap;        }        .child {          border: 2px dashed rgb(255, 0, 179);        }        #header {          grid-column: 1/4;        }        #footer {          grid-column: 1/4;        }        /\* To move any item vertically and horizontally center here we are making every item as a flexbox\*/        #parent > div {          display: flex;          justify-content: center;          align-items: center;        }      </style> | <body>      <div id="parent">        <div id="header" class="child">Header</div>        <div id="navbar" class="child">Navbar</div>        <div id="content" class="child">Content</div>        <div id="aside" class="child">Aside</div>        <div id="footer" class="child">Footer</div>      </div> |
|  | This example is with grid-template-areas and px  <style>        #parent {          display: grid;          background-color: rgb(239, 248, 110);          /\* grid-template-columns: 50px 50px 50px 50px;          grid-template-rows: 50px 50px 50px 50px; \*/          grid-template-columns: 50px 50px 50px 50px;          grid-template-rows: 50px 50px 50px 50px;          grid-template-areas:            "header header header header"            "navbar content content aside"            "navbar content content aside"            "footer footer footer footer ";          gap: 1cap;          justify-content: start;          /\* justify-items: center; \*/        }        .child {          border: 2px dashed orange;        }        #header {          justify-self: center;          grid-area: header;        }        #navbar {          grid-area: navbar;        }        #content {          grid-area: content;        }        #aside {          grid-area: aside;        }        #footer {          grid-area: footer;        }      </style> | <body>      <div id="parent">        <div id="header" class="child">Header</div>        <div id="navbar" class="child">Navbar</div>        <div id="content" class="child">Content</div>        <div id="aside" class="child">Aside</div>        <div id="footer" class="child">Footer</div>      </div> |
|  | //Without using child properties    <style>        #parent {          display: grid;          gap: 10px;          grid-template-columns: 10vw 10vw 10vw 10vw;          grid-template-rows: 10vh 10vh 10vh 10vh 10vh;          border: 2px dashed;        }        #header {          grid-column: 1/5;        }        #main {          grid-row: 3/4;          grid-column: 1/4;        }        #rightsidebar {          grid-row: 2/4;        }        #parent > div {          background-color: rgb(72, 230, 235);          /\* To make every ele content hor & ver          center make every ele as flexbox parent \*/          display: flex;          justify-content: center;          align-items: center;        }        #footer {          grid-column: 1/5;        }      </style> | <body>      <div id="parent">        <div id="header" class="child">Header</div>        <div id="sb1" class="child">smallbox</div>        <div id="sb2" class="child">smallbox</div>        <div id="sb3" class="child">smallbox</div>        <div id="rightsidebar" class="child">Sidebar</div>        <div id="main" class="child">Main content</div>        <div id="footer" class="child">Footer</div>      </div> |
|  |  |  |

Animations

For every animation it should have some name like (animation-name: **circle**;)

|  |  |
| --- | --- |
| animation-name | For every animation we should give some name & we have to apply that animation using @keyframes |
| animation-duration | For how much time each animation should run (**from** till **to** )  Ex:- animation-duration: 2s;  Here this single animation will take 2s to completely play that animation |
| animation-timing-function | In which style it should play the animation  Linear – same speed  Ease-in – delay in starting (starting slow ending fast)  Ease-out – delay in ending (first it will go very fast, later it will go slow)  Ease-in-out – delay fast delay (slow in starting, ending, middle is fast) |
| animation-delay | After how much time animation should start |
| animation-iteration-count | How many number of times it should play that animation -Any number / infinite  After its iteration count, no animation will be applied,  animation-iteration-count: 3;  means totally 3 times animation will be applied |
| animation-direction | Normal/reverse/alternate/alternate-reverse  Reverse means instead of happening from to🡪from, it will appear from from—to |
| animation-fill-mode | Forwards, backwards, both  Forwards – once animation is played at last –ele will settle in animation state instead of staying in original state  Backwards- after animation times –ele will come back to original state before animation (this is the default behaviour)  Both  Both , backwards both are same |
| animation-play-state | Paused,  Running  Ex:- make ele initially in paused state & on hover ..it will go to running state |
| Short hand function | Animation: <animation name> <how much time it should play> <style/ timing-function> <no of time it should play the animation>  <name> <entha sepu> <ela> <enni sarlu> <ayyaka ye state lo undali> |

These 3 properties are mandatory, we have to provide animation name

|  |  |  |  |
| --- | --- | --- | --- |
| #box {          background-color: rgb(150, 150, 150);          width: 100px;          height: 100px;          font-size: 5vw;          animation-name: **circle**;          animation-duration: 2s;    /\* count is how many times u want to see animation \*/          animation-iteration-count: 30;        }  After iteration times, animation will be off and it will go to normal state  Animation-delay: means applying animation after some delay | @keyframes <animation-name here> {   @keyframes **circle** {          /\* From will be the initial state \*/          from {            width: 300px;            height: 300px;            background-color: coral;          }          /\* To is the final animation state \*/          to {            border-radius: 50%;            width: 100px;            height: 100px;            background-color: yellowgreen;          }        } | Initial state    Final | Instead of from , to u can use 0% , 100%  0% means initial    0% {            justify-content: center;            height: 10vh;            font-size: 2vw;          }  50% means mid of total animation seconds          50% {            /\* justify-content: center; \*/            /\* border-radius: 50%; \*/            /\* font-size: 5vw; \*/          }  100% last sec of an animation          100% {            justify-content: flex-end;            height: 20vh;            font-size: 5vw;          } |

|  |  |
| --- | --- |
| animation-direction: normal | Always from  1st time from 0% -to 100%  2nd time also from 0% -to 100% |
| animation-direction: reverse | 1st time from 100% -to 0%  2nd time also from 100% -to 0% |
| Animation-direction: alternate | Means each time different navigation  1st time from 0% -to 100%  2nd time also from 100% -to 0%  3rd time from 0% -100%  4th time from 100%-0% |

|  |  |  |  |
| --- | --- | --- | --- |
| Demo for using percentages and animation directions | | | |
| **<style>**        #g1 {          display: grid;          grid-template-columns: 50% 50%;          /\* To see all ele in same row i placed in grid          where as each ele is a flex box\*/        }        #p11 {          /\*  since all ele needs to be in same row i used flex\*/          display: flex;          flex-direction: row;          flex-wrap: wrap;          background-color: rgb(221, 135, 104);          animation-name: fh;          animation-duration: 3s;          animation-iteration-count: 10;          animation-direction: alternate;        }        #p22 {          display: flex;          flex-direction: row;          background-color: rgb(197, 197, 30);          animation-name: sh;          animation-duration: 3s;          animation-iteration-count: 10;          animation-direction: alternate;        } | /\* Animation related properties \*/        @keyframes fh {          0% {            justify-content: center;            height: 10vh;            font-size: 2vw;          }          50% {            /\* justify-content: center; \*/            /\* border-radius: 50%; \*/            /\* font-size: 5vw; \*/          }          100% {            justify-content: flex-end;            height: 20vh;            font-size: 5vw;          }        }        @keyframes sh {          0% {            justify-content: center;            height: 10vh;            font-size: 2vw;          }          100% {            justify-content: flex-start;            height: 20vh;            font-size: 5vw;          }        }  **</style>** | <body>      <div id="g1">        <div id="p11" class="font-effect-fire-animation">          <div id="one">M</div>          <div id="two">A</div>          <div id="two">N</div>          <div id="two">I</div>        </div>        <div id="p22" class="font-effect-fire-animation">          <div id="three">S</div>          <div id="four">A</div>          <div id="four">N</div>          <div id="four">T</div>          <div id="four">U</div>        </div>      </div>      <div class="p2"></div>    </body> | Start of animation    End of animation |

Animation timing

|  |  |  |  |
| --- | --- | --- | --- |
| <style>        .box {          width: 200px;          height: 100px;          background-color: rgb(243, 110, 110);          text-align: center;          padding-top: 5vw;          box-sizing: border-box;          border: 2px dashed black;          margin-top: 50px;          font-size: 50px;          animation-duration: 5s;        }        #b1 {          animation-name: b1;          animation-iteration-count: 3;        }        #b2 {          background-color: rgb(223, 243, 110);          /\* animation-name: b2;          animation-iteration-count: 3;          animation-timing-function: ease-out;          animation-fill-mode: forwards;          animation-direction: alternate; \*/          /\* below is the shorthand operator  \*/          animation: b2 5s ease-out 10 reverse forwards running;        }        @keyframes b1 {          from {            transform: translateX(10%);          }          to {            transform: translateX(1000%);          }        }        @keyframes b2 {          from {            transform: translateX(10%);          }          to {            transform: translateX(1000%);          }        }      </style>    </head>    <body>      <div id="b1" class="box">Prakash</div>      <div id="b2" class="box">Inverters</div>    </body> | |  |  | | --- | --- | | Starting | ending | |
|  |  |