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> |
|  |  |

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

|  |  |
| --- | --- |
| Height  Ex:-   h3 {        background-color: darkmagenta;        color: white;        height: 500px;      }   <h3>See all the images</h3> | 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;      } |  |

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;

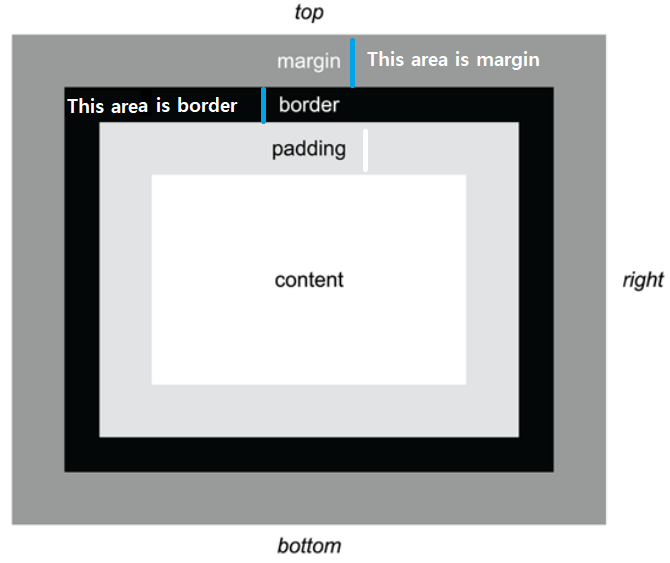
        }

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 \*/      } |

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 |

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) |

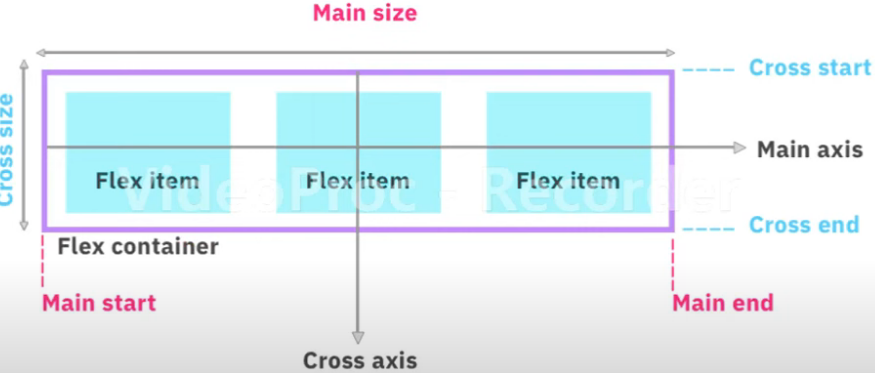
Z-Index

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



|  |  |
| --- | --- |
| flex-direction: row | Main axis will be horizontal (x-axis) |
| flex-direction: column | Then main axis will be vertical / Y-axis |

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

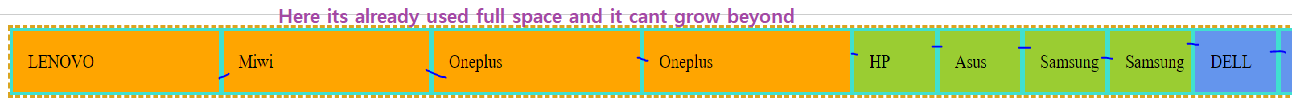
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| .parent {        display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;      } | .parent {       display: flex;        flex-direction: row;        flex-wrap: wrap;        justify-content: flex-start;  **align-content: 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

Always 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 \*/ |  |
| 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 am still confused the diff between   child’s   align-self: center; && parent’s align-content: center;