**Responsive layout**

**viewport** : Visible area of a browser window where a webpage can be displayed

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

Browser width = **device viewport size**

1 pixel on screen = one pixel on device independent screen = > **it should not be zoomed initially**

Should not use: user-scaleable

minimum-scale

maximum-scale

**Box Model**: Each HTML element is represented as rectangular box 🡪 padding, border, margin

Default box-sizing is content-box 🡺 **box-sizing:content-box**

Ex: div by default has padding, margin and border outside the div

So set ***box-sizing:border-box // padding, margin and border are added inside the div/box***

html{

box-sizing: border-box;

}

**\*,\*:before, \*:after**{

**box-sizing:inherit;**

}

**max-width:** max-width of an element to 100% allows it be viewed in any screen

**img**{

**max-width: 100%;**  // never be more wider than containing element

}

**display**: displays the type of box used to render an element

*display: inline;*

*display: block;*

*display: inline-block;*

*display: flex;*

*display: grid;*

**position:** used to change the position of an element on the page

position: **static**; default – **flow of the page** – **no left, right, bottom, top properties**

position: **relative;** moves the element in **relation to its original location**

**doesn’t affect other elements, leaves empty spaces** on its original position

position: **absolute**; moves element in **relation to the container element** (div, body etc.) that doesn’t have a position of static

***doesn’t leave empty spaces****, it overlaps*

***doesn’t have default width***

position: **fixed**; positions element in **relation to the whole page** and it **stays in that spot**

position: **sticky**; element is placed on the **position as normal and when scrolled sticks to the specified position**

Z-index: **to position overlapping element**: **z-index: 1 /auto /-1** //works only for positioned elements

**Float :** places an element along the left side or right side of its container, allowing other elements to wrap around it.

Works well with image and text around

.photo{

**float:left;**

}

.paragraph{

**Clear:left; // clear:right // clear:both** // to display a paragraph not wrapping after float

}

**Units:**

* **pixels : px** // smallest controllable point of color on a screen // fixed size
* **percent :** **Relative units** – length can vary – based on height or width of the containing element

for responsive contents **p{ width:50%}**

* **vw** -viewport width : relation to viewport – 1/100th of the viewport width
* **vh** -viewport height: 1/100th of the viewport height – **50vh – half of the viewport**
* **v-min –** becomes small if any one (h or w) becomes small
* **v-max** – enlarges if any one (h or w ) increases

div{

width:50vmin;

height:50vmin;

}

* **fr** : fractional unit – allowed only in grid layout

**Media Query**

Used to modify the CSS of a site depending on qualities such as viewport size or device type

**@media:** applies one or more CSS properties based on the result of a media query

**@media(min-width:600px**){ // ***@media*** *followed by the* ***query***

body {

background-color:gray;

}

}

**@media(max-width:320px)**

**Accessibility:**

Tab-order: see to that the tab-key when pressed in keyboard selects elements in order

**Grid Layout:** Grid based layout system (two directions)

**display:grid**; Tells browser it’s a grid element and every child element inside it a grid item

// ***developer tools - >inspector-> layout : grid details are listed***

***grid-template-columns:200px 100px; //*** *defines the columns of the grid*

***grid-template-rows:200px 100px; //*** *defines the rows of the grid*

***grid-gap:10px 20px****;* ***grid-gap:20px;***  *// gap between rows and columns*

***grid-column-gap: 50px;*** *// using margin property will effect the ends of the grid*

***grid-row-gap:25px;***

**grid-template-columns & grid-template-rows**

**grid-template-columns:auto auto auto;**

**grid-template-columns:auto 100px auto;**

**fr : a unit of measurement based on a fraction of unused space in the grid container**

**grid-template-columns:1fr 100px 1fr;** 1fr /2fr and 1fr/2fr

**grid-template-columns:1fr 100px 2fr;** 1fr /3fr and 2fr/3fr

**grid-template-columns:1fr 3fr 9fr; //**its not going to split as planned because sometimes fr’s may adjust itself to content and other fr’s are allocated according

**grid-template-rows :2fr 1fr 3fr;**

**height:90vh // containers height if height is fixed the row fr’s will be divided according to the height of the container or else it will be divided according to the contents size**

**grid-template-columns:repeat(4,125px);**

**grid-template-columns:repeat(2,70px 1fr);**

**Placing grid items :**

**grid-column-start: 2;** // specifies the grid starting and ending column

**grid-column-end:4; // can view developer environment to view col numbers**

**grid-column: 2 /4;** //single line to specify start and end columns

**grid-column-end:-1;** // specify negative values – to start from the column end

**grid-column: 2 / span 2;** // start from 2 and span for 2 columns

**grid-row: 3 / span 2; // if starting and ending values of grid item is specified it will go exactly at that location and overlaps (use z-index) even if the location has other grid items , if just row or column value is specified it finds the next available free space to place the grid items**

**grid-area** :**grid-row-start,grid-column-start,grid-row-end,grid-column-end;**

**Grid alignment :**

**justify-content:** positions columns horizontally

*Start(default), end ,center ,space-around, space-between, space-evenly*

**align-content:** positions rows vertically || **only if height is provided ex: min-height: 90vh**

*Start(default) , end ,center ,space-around, space-between, space-evenly*

**justify-items:** positions grid items horizontally within columns

*Start , end ,center, stretch(default)*

**align-items:** positions grid items verticallywithin each row

*Start , end ,center, stretch(default)*

**justify-self:** individual items || horizontally

*Start , end ,center, stretch(default)*

**align-self:** individual items || vertically

*Start , end ,center, stretch(default)*

**Variable Columns:** responsive layout

If Not sure how many columns because it depends on view port

**grid-template-columns:repeat(auto-fill,150px);**

**grid-template-columns: repeat (auto-fill, minmax(100px,1fr)); // min 100px but if space is available it can be upto 1fr width || to save unused space between items**

**justify-content: space-between;** // to align columns to occupy spaces in the container

**grid-template-columns: repeat (auto-fit, minmax(100px,1fr)); //auto fit does not produce empty columns when number of items is very less**

**grid-template-columns:50px minmax(100px,1fr) minmax(150px,1fr); //** when on minimum size the min values of columns are applied and on max it is equally split

**grid-template-areas:** defines areas of the grid that can be referenced **by name**

**grid-template-areas:**

"mango orange roseberry"

"banana banana rasberry"

" grapes strawberry" //**dot for cells without name**

;

// no of names in every row should be the same. // should follow rectangle area

.item20{

grid-area:orange;

}

// item 20 will be displayed in the cell which has name orange

**Naming grid lines:**  line names to position grid items

**grid-template-columns: [start] 1fr [content-start] repeat(3,1fr) [content-end] 1fr [end];**

.item19{

**grid-column:start / end;**

}

**Order:** to change the order of the items

.item20{

**order:4;** // order:-3 // default value is 0 for all items

}

**Grid flow and implicit tracks:**

**grid-auto-rows: 200px;**

**grid-auto-flow: column dense;**

**grid-auto-columns: 1fr;**

**Flexbox: web items layout system (one direction)**

One dimensional layout system in css

**It can be divs / ul / any other list of items**

**display: flex;**

min-height:50vh; // flex items will occupy the whole height of the container

**flex-directions:**

**horizontal** (main axis) : left-right **vertical (**cross-axis) : top - bottom

right- left bottom - top

**flex-direction:row;**

**flex-direction:row-reverse;**

**flex-direction:column;**

**flex-direction:column-reverse;**

**flex-flow:row-reverse wrap;**

**flex-wrap:**

**flex-wrap : nowrap;**

**flex-wrap : wrap;**

**flex-wrap:wrap-reverse;**

**order:** changes the order of the flex item in the container

.item{

**Order:3;** // same as grid order

}

**Flexbox alignment:**

**justify-content:** positions items horizontally

flex-start, flex-end ,center ,space-around, space-between, space-evenly

**align-content:** to align multiple rows of items || **only to wrapped flex items** flex-start, flex-end ,center ,space-around, space-between, space-evenly

**align-items:** aligns items cross axis

flex-start , flex-end ,center, stretch, baseline( when items are of different sizes –say font sizes)

**align-self:** individual items || vertically || cross-axis

flex-start, flex-end ,center, stretch, baseline

**Distributing space to flex items: //**similar to fr in grid

**Flex-grow:** gives flex items the ability to grow if there is space and allows distribution of spaces || **unit less number** || **should be +**

.item3{ .item5{

**flex-grow: 1;**  flex-grow:2; //like frame 2:1

} }

**Flex-shrink:** gives flex item the ability to shrink if space is needed

.item3{

**flex-shrink:3;**  // item3 will shrink first if space is not available }

**Flex-basis:** defines the default size of an element **// works even if horizontal | vertical**

.item{

Flex-basis:150px;

}

**Flex:** combination of flex-grow,flex-shrink and flex-basis

.item3{

**Flex: 2 1 auto; //**grow 2 shrink 1 and size auto

}

**Layout Design:**

**Centering an element:** // ex: container div has an element div

**.container { .container {**

**display:flex; display:grid;**

**justify-content:center; justify-content:center;**

**align-items:center; align-content:center;**

**} }**

**3-column layout with grid areas:**

Ex: container with divs for header , article ,related links ,ads ,footer

**.container{**

**Display:grid;**

**Grid-gap:20px; // gap between grid items**

**Grid-template-areas:**

**“header”**

**“content” // creating name for each grid items**

**“related”**

**“ads”**

**“footer”**

**}**

**.main-header{**

**Grid-area:header; // assigning class/div with grid name**

**}**

.**main-content{**

**@media (min-width: 500px){**

**.container{**

**Grid-template-columns:repeat(3,1fr);**

**Grid-template-areas:**

**“header header header”**

**“content content related”**

**“content content ads”**

**“footer footer footer “**

**}**

**}**

**@media (min-width: 800px){**

**.container{**

**Grid-template-columns:repeat(4,1fr);**

**Grid-template-areas:**

**“header header header header”**

**“related content content ads”**

**“footer footer footer footer “**

**}**

**}**

**Grid-area:content;**

**}**

**.related-links{**

**Grid-area:related;**

**}**

**.adds{**

**Grid-area:ads;**

**}**

**.main-footer{**

**Grid-area:footer;**

**}**

**3-column layout with flex-box:**

Ex: it has nav, container with related-contents, content, add and footer

**@media (min-width:500px){**

**Display:flex;**

**}**

**.main{**

**Order:2;**

**flex-grow:2;**

**}**

**.related{**

**Order:1;**

**flex-grow:2;**

**}**

**.adds{**

**Order:3;**

**flex-grow:2;**

**}**

**12-column layout setups**

Ex: it has container with elements for header, nav, main-content, links, ads, footer

**@media(min-width:900px){**

**.links{**

**grid-column: 1 /span 2;**

**}**

**.content {**

**grid-column:3/ 7;**

**}**

**.adds{**

**grid-column:10 / span 3;**

**grid-row:3;**

**}**

**.footer{**

**grid-column: 1/ span 12;**

**}**

**}**

**.container{**

**max-width:1200px;**

**}**

**// whole container will be 1200px even if the user uses very very wide screen (don’t want to go on which will not be nice)**

.container{

display:grid;

grid-template-columns:repeat(12,1fr);

grid-gap:10px;

}

nav ul{

list-style:none;

margin:0;

padding:0;

}

**.container > \*{** **//all the child element of container**

Grid-column : 1 / span 12;

**}**

**@media(min-width:600px){**

**.links{**

**grid-column: 1 /span 3;**

**grid-row:3;**

**}**

**.adds{**

**grid-column:1 / span 3;**

**grid-row:4;**

**}**

**.content, .footer{**

**grid-column:4 / 9;**

**}**

**nav{**

**display:flex;**

**justify-content:space-between;**

**}**

**}**