CSS: Cascading Style Sheets

Style to our web pages. Make websites responsive.

Syntax:

Selector { prop: val }

P { color: blue }

3 ways to add CSS to markup:

1.Inline CSS: add CSS to element directly using style attribute.

<p style=”color:blue;background-color:yellow”>Hello world </p>

2.Internal CSS: kept CSS inside header tags in <style> tag.

<style>

P {

color: blue;

background-color: yellow

}

</style>

3.External CSS: add CSS in .css file and include .css file in markup.

To include css file use link tag.

<link rel=”stylesheet” href=”style.css”>

Use **! important** to avoid overriding of CSS.

CSS Selector: easily target single/multiple html elements in markup.

Element selector: use element name to select.

Id selector: use **#id\_name** to select.

Class selector: use **.class\_name** to select.

Group selector: use , between multiple names of selector.

Comments syntax: /\* comment is here \*/

div p: Selects all <p> elements **inside** <div> elements.

div>p: Selects all <p> elements where the **direct** **parent** is a <div> element.

div + p: Selects all <p> elements where the **previous sibling** is a <div> element.

li:nth child(even|odd|2n+0) : selects all even, odd, expression list item.

Input[type=’text’]: selects input elements with type as text.

a[target=’\_blank’]: select an element with target as blank.

The ::before selector inserts something before the content of each selected element(s).

Use the ::after selector to insert something after the content.

tag::before {

css declarations;

}

border: border-width border-style border-color;

eg.border:4px solid green;

border-top: border of top side of block.

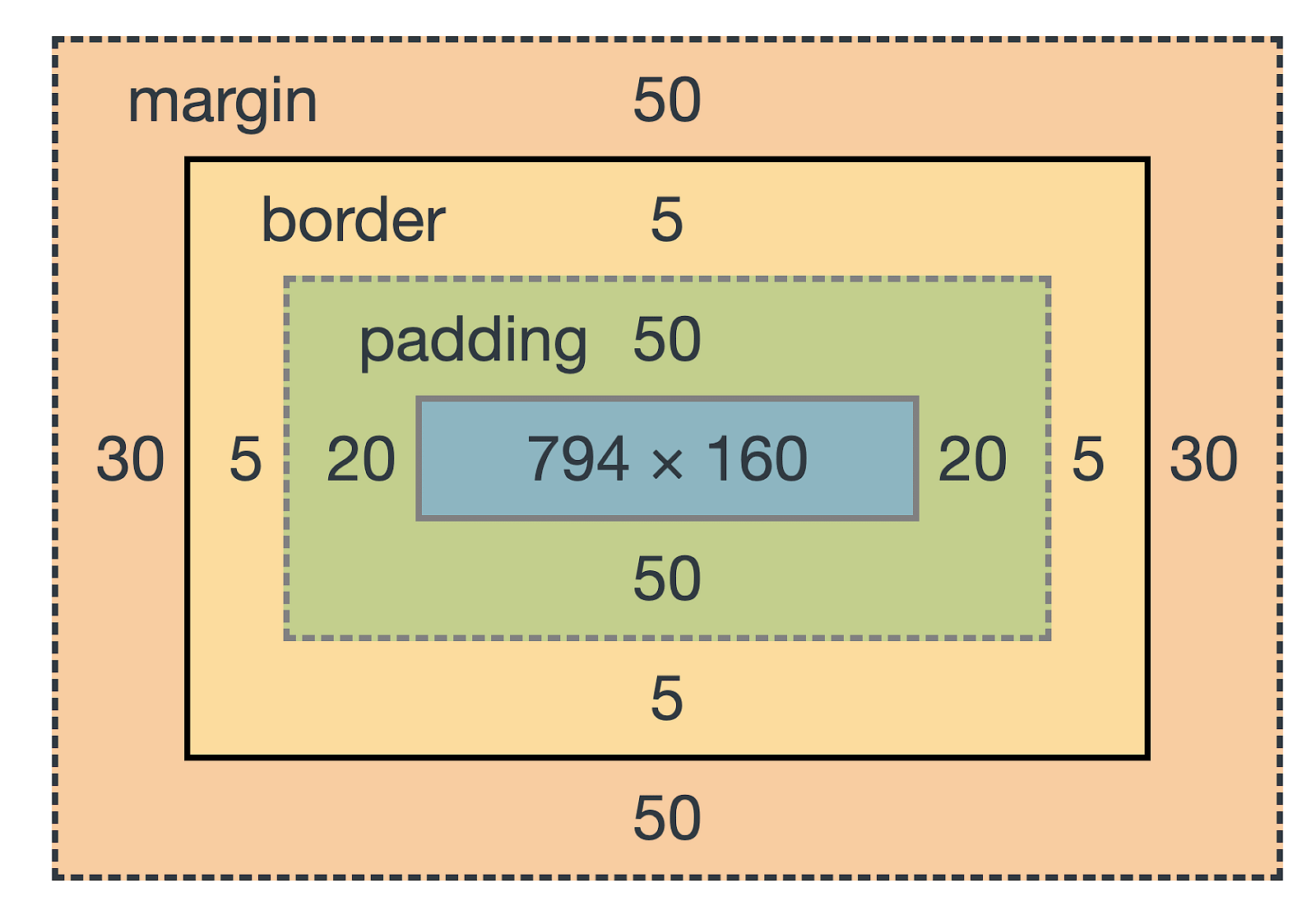
border-radius: corner radius.

background-image:url(‘ ’);

background-repeat: no-repeat // show img only one time.

Background-position: center center;

Box Model:



padding: top right bottom left;

padding:(top-bottom) (right-left);

margin syntax is same as padding.

box sizing: border box ; // allows us to include padding and border in an element's total width & height.

margin: auto; // div element at center horizontally.

float & clear:

The CSS float property specifies how an element should float.

When we use the float property, and we want the next element below (not on right or left), we will have to use the clear property.

display:

inline-block element does not start on a new line, but it takes up the specified width and height.

A block element starts on a new line and occupies available width or its specified width.

Inline element will not start on a new line, will only take up as much width as the content it contains.

position: static|absolute|fixed|relative|sticky;

after position property we can use top, bottom, left, right property.

static : Default value. Elements render in order, as they appear in the document flow.

relative : The element is positioned relative to its normal position.

absolute : The element is positioned relative to its parent element position.

fixed: element stays fixed to its position irrespective of happening on screen it is fixed at the viewport.

Sticky: The element is positioned based on the user's scroll position

The **z-index** property specifies the stack order of an element. It will work on positioned element.

An element with greater stack order is always in front of an element with a lower stack order.

The visibility property specifies element is visible or not.

visibility: hidden; //element is hidden but space is reserved.

display: none; // element is hidden & space is not reserved.

box-shadow: none|h-offset v-offset blur spread color;

inset: Changes the shadow from an outer shadow (outset) to an inner shadow.

text-shadow: h-offset v-offset blur spread color;

Custom variable and Properties: update one value and have it reflected in multiple places.

--spacing: 1.2rem; //declaring var

padding: var(--spacing); //use var

to make global var: :root{ --spacing: 1.2rem; }

Animations:

An animation lets an element gradually change from one style to another.

It is also possible to use percent in keyframes.

@keyframes example {

from {background-color: red;}

to {background-color: yellow;}

}

**animation-name**: example; name of animation

**animation-duration**:2s; property defines how long an animation should take to complete.

**animation-delay**: 2s; a delay for the start of an animation.

**animation-iteration-count**: 3; number of times an animation should run. Use **infinite** to run forever.

**animation-direction**: normal|reverse|alternate|alternate-reverse;

alternate: forward-backward cycle; alternate-reverse: backward-forward cycle.

**animation-timing-function**: property specifies the speed curve of the animation.

ease - slow start, then fast, then end slowly (this is default).

linear -same speed from start to end.

ease-in - slow start.

ease-out - slow end.

ease-in-out - slow start and end.

**animation-fill-mode** property specifies a style for the target element when the animation is not playing.

forwards: end on to. backwards: end on from.

**Animation Shorthand Property :**

**animation**: name duration timing\_function delay count direction.

**CSS Transitions:** CSS transitions allows you to change property values smoothly, over a given duration.

**transition:** **transition-property transition-duration transition-timing-function transition-delay;**

CSS **transforms** allow you to move, rotate, scale, and skew elements.

translate(x,y) method moves an element from its current position.

rotate(120deg) method rotates an element clockwise or anti-clockwise.-ve value rotate anticlockwise.

scale() method increases or decreases the size of an element.

transform: scale(2, 3); increase width twice and height thrice.

scaleX(2) for width; scaleY(3) for height.

skewX(deg) & skewY(deg) method skews an element along the X-axis& Y-axis by the given angle.

Media Query: To make website responsive.

@media screen and (min-width: 480px) { ……………………………… }

CSS Grid:

Display:Grid in container

The vertical lines of grid items are called columns.

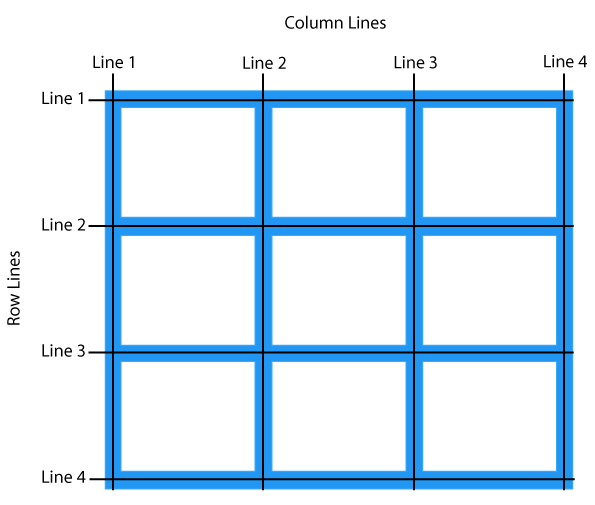
The horizontal lines of grid items are called rows.

The spaces between each column/row are called gaps.

row-gap: 50px;

column-gap: 100px;

gap: 50px 100px; gap: row gap column gap



grid-template-columns Specifies the size of the columns, and how many columns in a grid layout.

grid-template-columns: 200px 100px 200px 100px;

grid-template-columns: repeat(2,200px 100px) ;

grid-template-rows Specifies the size of the rows, and how many rows in a grid layout.

**The grid-column** property specifies a grid item's size and location in a grid layout.

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

grid-column: 2/ 3;

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

use the **span** keyword to avoid specifying end lines. span means number of block.

.item {grid-column: 2 / 5; grid-row: 1 / 3;}

.item { grid-column: 2 / span 3; grid-row: 1 / span 2;}

-1 value use to occupy till end. grid-column: 2 / -1; total 5 column line.it will drag till last.

manually defined grid is called the explicit grid.

The grid-auto-rows and grid-auto-columns properties give us control over the size of implicit tracks.

grid-auto-columns: 200px;

grid-auto-rows: 60px;

Implicit tracks will now always have a width of 200px and a height of 60px, no matter if the grid item fits or not.

grid-auto-flow: column;

We can specify how auto-placed items get flowed into the grid by using the grid-auto-flow property.

***Justify property use to move horizontally and align property to move vertically.***

Justify-content control the alignment of grid column. i.e. div

Justify-item control the alignment of grid item. Example <p>, <h1>

Justify-self override the property of justify-item.

the **min**-**content** uses the length of the widest bit of content in the element box.

**max**-**content** represents the size a box needs to contain all its content without being wrapped or it overflows the box.

minmax(min max) Sets a size range greater than or equal to min and less than or equal to max.

auto-fit keyword will expand the grid items to fill the available space. auto-fill will keep the available space reserved without altering the grid items width.

CSS FLEXBOX:

The Flexible Box Layout Module makes it easier to design flexible responsive layout structure without using float or positioning.

The flex container becomes flexible by setting the display property to flex.

**display: flex; // in on line**

The flex container properties are:

**flex-direction**: direction of the container flex items.

flex-direction: row |column|row-reverse|column-reverse

The **flex-wrap** property specifies whether the flex items should wrap or not.

flex-wrap: wrap| wrap-reverse;

**flex-flow** property is a shorthand property for setting both the flex-direction and flex-wrap properties.

flex-flow: row wrap;

The **flex** property sets the flexible length on flexible items.

flex:1|2|auto|50%

The **justify-content** property is used to align the flex items horizontally.

justify content: flex-start|flex-end|center|space-around|space-between|space-evenly

space-around: equal space on both side

space-between: no corner space

space-evenly: equal space with corner space.

Align-items: flex-start|flex-end|center // vertical position of the item.

The direct child elements of a flex container automatically become flexible (flex) items.

The **order** property specifies the order of the flex items. Eg order:1

The **flex-grow** property specifies how much a flex item will grow relative to the rest of the flex items.

The **flex-shrink** property specifies how the item will shrink(reduced) relative to the rest of flexible items.

The **flex-basis** property specifies the initial length of a flex item. flex-basis: 200px

If direction -row then control width. if direction-column then control height.

**Flex: grow shrink basis;**