

CSS Notes



[codingwithyash](#)



codingwithyash.com

Introduction to CSS

→x →x →x →

HTML is just an skeletal layout of a website. We need CSS to design a website, add styles to it and make it look beautiful.

Q What is CSS?

CSS stands for cascading style sheet. CSS is optional but it converts an off looking HTML page into a beautiful & responsive website.

Q Why learn CSS?

CSS is a very demanded skill in the world of web development. If you are successfully able to master CSS, you can customize your websites as per your liking.

Q What is DOM?

DOM stands for document object model. When a page is loaded, the browser creates a DOM of the page which is constructed as a tree of objects.

(*) HTML id and class attributes.

When an HTML element is given an id, it serves as a unique identifier for that element.

On the other hand, when an HTML element is given a class, it now belongs to that class. More than one elements can belong to a single class but every element must have a unique id (if assigned).

We can add multiple class to an element like this

`<div id = "first" class = "c1 c2 c3">`

↓
unique ID

↓
multiple class followed by spaces.

Three ways to add CSS to HTML.

- ① **Inline CSS** → Adding CSS using style attribute to that element.
- ② **Internal CSS** → Adding `<style>..... </style>` to HTML.
- ③ **External CSS** → Adding a stylesheet (.css) to HTML using `<link>` tag.

* CSS Selectors

A CSS selector is used to select an HTML element for styling.

ex → `body` { → Selector

`color: red; → Declaration (Property: value;)`
`background-color: pink;`
`}`

Element Selector

— x — x — x —

It is used to select an Element based off the tag name.

ex →

`h2` {

`color: blue;`
`font-size: 46px;`
`}`

Id Selector

— x — x —

It is used to select an element with a given id.

ex →

`#first` {

`color: white;`

`background-color: black;`

`}`

is used to target by id.

class selector

It is used to select an element with a given class.

ex ->

```
.med {  
  background-color: red;  
}
```

Note:

① We can group selectors like this:

h1, h2, h3, div {

color: blue;

}

② We can use element.class as a selector like this:

p.med {

color: red;

}

all paragraph of med class will get red color.

③ * (asterisks) can be used as a universal selector to select all the elements.

* {

margin: 0;

padding: 0;

}

④ An inline style will override external and internal styles.

Colors & Backgrounds

CSS rules are simple key-value pairs with a selector. We can write CSS rules to change color and set backgrounds.

The color property

The CSS color property can be used to set the text color inside an element.

ex →

```
p {  
  color: red;  
}
```

Text color will be changed to red.

Similarly we can set color for different elements.

Types of color values

Following are the most commonly used color values in CSS.

- ① RGB → specify color using Red, green, blue value e.g. `rgb(200, 90, 70)`
- ② HEX code → specify color using hex code. e.g. `#ff7180`
- ③ HSL → specify the color using HSL values. e.g. `hsl(0, 90%, 63%)`
Hue, saturation, lightness

The value of the color or background color is provided as any of these values.

Note: We also have an RGBA and an HSLA values for color but they are rarely used by beginners.

A stands for Alpha.

④ Favicon → `<link rel="icon" href="favicon.ico?v=2">`
this is optional.

The background-color property:

The CSS background-color property specifies the background color of a container.

for eg.

```
.brown {
```

```
background-color: brown;
```

```
}
```

Sets the background color of brown class to brown.

The background image property

used to set an image as the background.

```
body {
```

```
background-image: url("yash.png");
```

```
}
```

The image is by-default repeated in x & y directions.

The background-repeat property:

can be any of:

① repeat-x → repeat in horizontal direction

② repeat-y → repeat in vertical direction

③ no-repeat → image not repeat.

The background-size property:

can be following

① cover → fits and no empty space remains

② contain → fit & image is fully visible

③ auto → Display in original size.

④ `{{width}}` → set width and height will be set automatically.

⑤ `{{width}}` `{{height}}` → set width and height.

The background-position property;

Sets the starting position of a background image.

```
.div1 {
```

```
background-position: left top;
```

```
}
```

The background-attachment property

Defines a scrollable/non-scrollable character of a background image.

```
.div2 {
```

```
background-attachment: fixed;
```

```
}
```

The background shorthand

A single property to set multiple background properties.

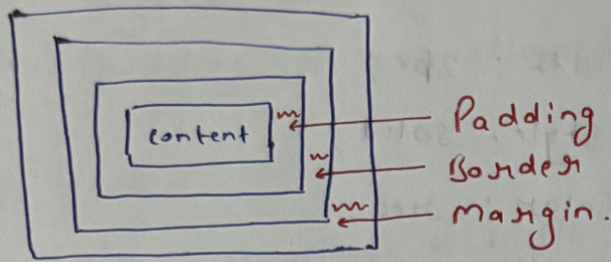
```
.div3 {
```

```
background: url("yash.png") no-repeat contain fixed  
right top;
```

one of the properties can be missing given the others are in order.

CSS Box model

The CSS box model looks at all the HTML elements as boxes.



Setting width & Height.

We can set width and height in CSS as follows.

```
#box {  
  height: 70px;  
  width: 70px;  
}
```

Note that the total width/height is calculated as follows:

Total height = height + top/bottom padding + top/bottom border + top/bottom margin.

⊛ Setting margin and padding.

```
.box {  
  margin: 3px;  
  padding: 4px;  
}
```

```
.box {  
  margin: 7px 0px 2px 11px;  
}
```

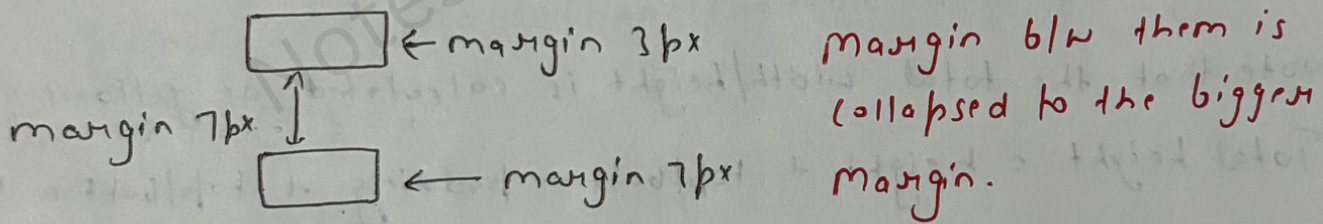
```
.box {  
  margin: 7px 3px;  
}
```


* Setting borders

We can set the border as follows.

```
• box {  
  border-width: 2px;  
  border-style: solid;  
  border-color: red;  
  border-radius: 7px;  
}
```

Margin collapse → when two margins from different elements overlap, the equivalent margin is the greatest of the two. This is called margin collapse.



Fonts and Display

The display property: The CSS display property is used to determine whether an element is treated as a block/inline element & the layout used for its children.

↳ flexbox/grid etc.

⊗ `display: inline` →

Takes only the space required by the element. No line breaks before and after. Setting width/height or margin/padding is not allowed.

⊗ `display: block` →

Takes full space available in width and leaves a newline before and after the element.

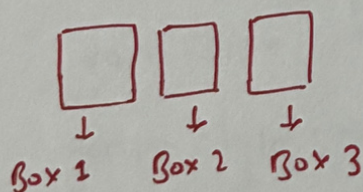
⊗ `display: inline-block` →

Similar to inline but setting height, width, margin and padding is allowed. Elements can sit next to each other.

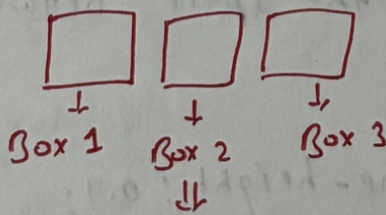
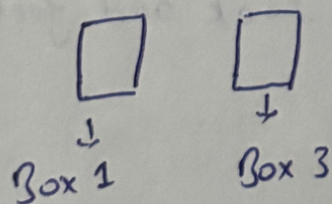
⊗ `display: none` vs `visibility: hidden` →

With `display: none`, the element is removed from the document flow. Its space is not blocked.

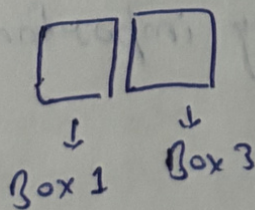
With `visibility: hidden`, the element is hidden but its space is reserved.



`visibility: hidden;`



`Display: none;`



⊗ Text-align property: →
used to set the horizontal alignment of a text.

```
div1 {  
    text-align: center;  
}
```

⊗ Text-decoration property: →

used to decorate the text
can be overline, line-through, underline, none...etc.

⊗ text-transform property →

used to specify uppercase and lowercase letters in a text.

```
p. uppercase {  
    text-transform: uppercase;  
}
```

⊗ line-height property →

used to specify the space between lines.

```
small {  
    line-height: 0.7;  
}
```

⊗ font → font plays a very important role in the look and feel of a website.

⊛ Font-family: → font family specifies the font of a text can hold multiple values as a "fallback" system.

```
p {  
  font-family: "times new roman", "monospace";  
}
```

↳ always do this to ensure the correct font of your choice is rendered.

Web safe fonts: → These fonts are universally installed across browsers.

⊛ How to add google fonts → In order to use custom google fonts, go to google fonts then select a style and finally paste it to the style.css of your page.

⊛ Other font properties: → Some of the other font properties are listed below:

font-size: → sets the size of the font

font-style: → sets the font style.

font-variant: → sets whether text is displayed in small-caps.

font-weight: → sets the weight of the font.

Font-family can be specific or it can be generic. ← [Research]

Size, position and lists

There are more units for describing size other than 'px'. These are rem, em, vw, vh, percentages etc.

Q What's wrong with pixel?

Pixels are relative to the viewing device. For a device with size 1920×1080 , 1px is 1 unit out of $1080/1920$.

* Relative lengths. $\rightarrow$ These units are relative to the other length property.

Following are some of the most commonly used relative lengths.

- ① em $\rightarrow$ unit relative to the parent font size.
 $\hookrightarrow$ em means "my parent element's font size".
- ② rem $\rightarrow$ unit relative to the root font size (`<html>` tag)
- ③ vw $\rightarrow$ unit relative to 1% viewport width.
- ④ vh $\rightarrow$ unit relative to 1% viewport height.
- ⑤ % $\rightarrow$ unit relative to the parent element
- ⑥ min/max - height/width property $\rightarrow$
css has a min-height, max-height, min-width and max-width property.

If the content is smaller than the minimum height, minimum-height will be applied.

similar is the case with other related properties.

(*) The position property :-

Used to manipulate the location of an element.

following are the possible values :

static → The default position top/bottom/left/right/z-index has no effect

relative → The top/bottom/left/right/z-index will now work. otherwise the element is in the flow of document like static. अपनी पुरानी position में।
अब Parent position में gap में मिलेगा।

absolute → The element is removed from the flow & is relatively positioned to its first non-static ancestor. top/bottom etc works.
Relative to the Parent.

fixed → Just like absolute except the element is positioned relative to the browser window.

sticky → The element is positioned based on user's scroll position.

(*) The list-style property:

The list style property is a shorthand for type, position, & image.

उदा.

list-style : square inside url ('http://img')

2-index property → It will only work when the position is not static.

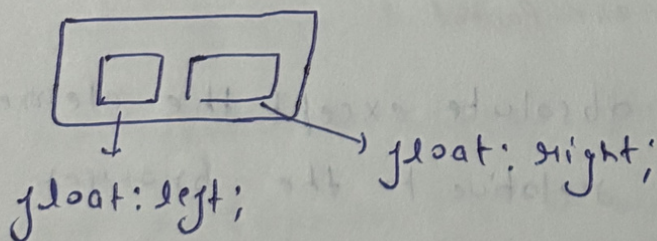
The 2-index property specifies the stack order of an element.

It defines which layer will be above which in case of overlapping elements. It works only for position → fixed, absolute, relative.

flexbox → m.m. imp.

Before we look into CSS flexbox, we will look into float and clear properties.

The float property → float property is simple. It just flows the element towards left/right



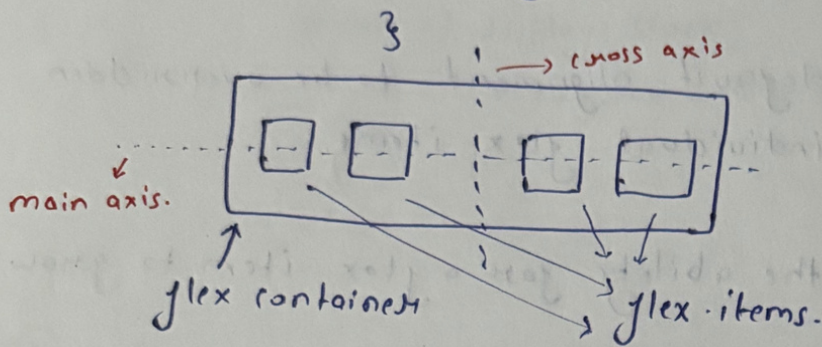
*) The clear property → used to clear the float. It specifies what elements can float beside a given element.

*) The CSS flexbox → Aims at providing a better way to layout, align and distribute space among items in a container.

• container {

display: flex;

→ Initialize a flexbox.



⊛ flex-direction property :-

Defines the direction towards which items are laid.

Can be row, row-reverse, column, column-reverse,

↓
default.

⊛ flex properties for parent (flex container).

following are the properties for flex parent:

① flex-wrap :- can be wrap, no-wrap, wrap-reverse, wrap items as needed with this property

② justify-content :- Defines alignment along main axis.

③ align-items :- Defines alignment along cross axis.

④ align-content :- Aligns a flex container's lines when there is extra space in the cross axis

⊛ flex properties for the children (flex items).

following are the properties for the flex children.

- ① **order**: controls the order in which the item appear in flex container.
- ② **align-self**: Allows default alignment to be overridden for the individual flex items.
- ③ **flex-grow**: Defines the ability for a flex item to grow
- ④ **flex-shrink**: specifies how much a flex item will shrink relative to the rest of the flex items.

⊛ **Media queries** → A way to make our design responsive when we resize our window.

Note → to create multiple div with ids use this shorthand.

div.box #box-\$*4

↓ ↓
for class for id

Output → `<div class="box" id="box-1"> </div>`

`<div class="box" id="box-2"> </div>`

`<div class="box" id="box-3"> </div>`

`<div class="box" id="box-4"> </div>`

Syntax for media screen: →
query

@media only screen^{and} (max-width: 300px) {
 #box-1 { display: block; }
}

→ It represent that we are using only "only screen" media query.

```
@media only screen (max-width: 500px) and (min-width: 300px)
{
  div-box-2 { display: block;
               background-color: blueviolet;
             }
}
```

(*) More on CSS selectors :-

- ① $div > p \Rightarrow$ If p is right inside div .
↳ This greater than symbol is used to target the direct child.
- ② $div + p \Rightarrow$ If p is right after div .
↳ This plus symbol is used to target the next sibling.
- ③ for selecting using attributes.

ex -> `input [type = 'text']` it is an input and target
{
 type = text
}

ex -> `a[target]` it is an anchor tag and select only
{
 target attribute
}

④ targeting n^{th} child.

ex → `li:nth-child(3)` at nth child me Target me at
 {
 }
 }
 k is use in this way

ex → `li:nth-child(2n+0)` 2 even it apply it in 2nd css
 {
 }

* Before and After pseudo selectors.

Syntax: `header::before {`
`background: url('images/tile1-1.png');`
`background-size: cover; center center; cover;`
`content: " ";`
`position: absolute;`
`top: 0; left: 0;`
`width: 100%;`
`height: 100%;`
`z-index: -1;`
`opacity: 0.6;`
`}`

Practise

* Box shadow and text shadow.

Syntax: `box-shadow: 10px 13px green;`
 ↓ ↓ ↓
 for x for y color.

These no. can be negative also
 In case of negative shadow upon choli Jaegi.

Syntax: 7px 5px 10px green;
 ↓ ↓ ↓ ↓
 font x font x blur radius color.

Syntax: 7px 5px 10px 2px green;
 ↓ ↓ ↓ ↓ ↓
 font x font x blur spread radius color.

* CSS Variables

we can declare two types of variable ① Global
 ② Local.

Global variable

Syntax:

```
:root {
```

```
--primary-color: blue;  
--danger-color: red;  
--maxw: 333px;  
}
```

These variables
 → can be used
 anywhere.

Local variable

```
.box1 {
```

4e variable mññ ← --box-color: violet;

box1 class nññ ññ
 accessible ññ /

```
}
```


(*) CSS animations and keyframes

• box 1

{

background-color: green;

width: 250px

height: 250px

position: relative

Basic &
mandatory for
animation

animation-name: Yash;

name could be anything.

animation-duration: 2s;

animation-iteration-count: 1;

animation-fill-mode: alternate;

animation-timing-function: ease-in;

}

animation @keyframes Yash {

0% {

from {

width: 200px;

50% {

to {

width: 400px

100% {

}

}

ଶୁଭି ଅମିତ

@keyframes Yash {
0% {

top: 0px;
left: 0px

}

25% {

top: 250px;
left: 0px;

}

75% {

top: 250px;
left: 250px;

}

100% {

top: 0px;
left: 250px;

}

}

⊛ CSS transitions → keyframes and animation.

→ To animate an element, you need to know about the animation properties and the @keyframes rule.

The animation properties control how the animation should behave and the @keyframe rule controls what happens during the animation.

There are eight animation properties in total.

Thank You !!!



[codingwithyash](#)



codingwithyash.com