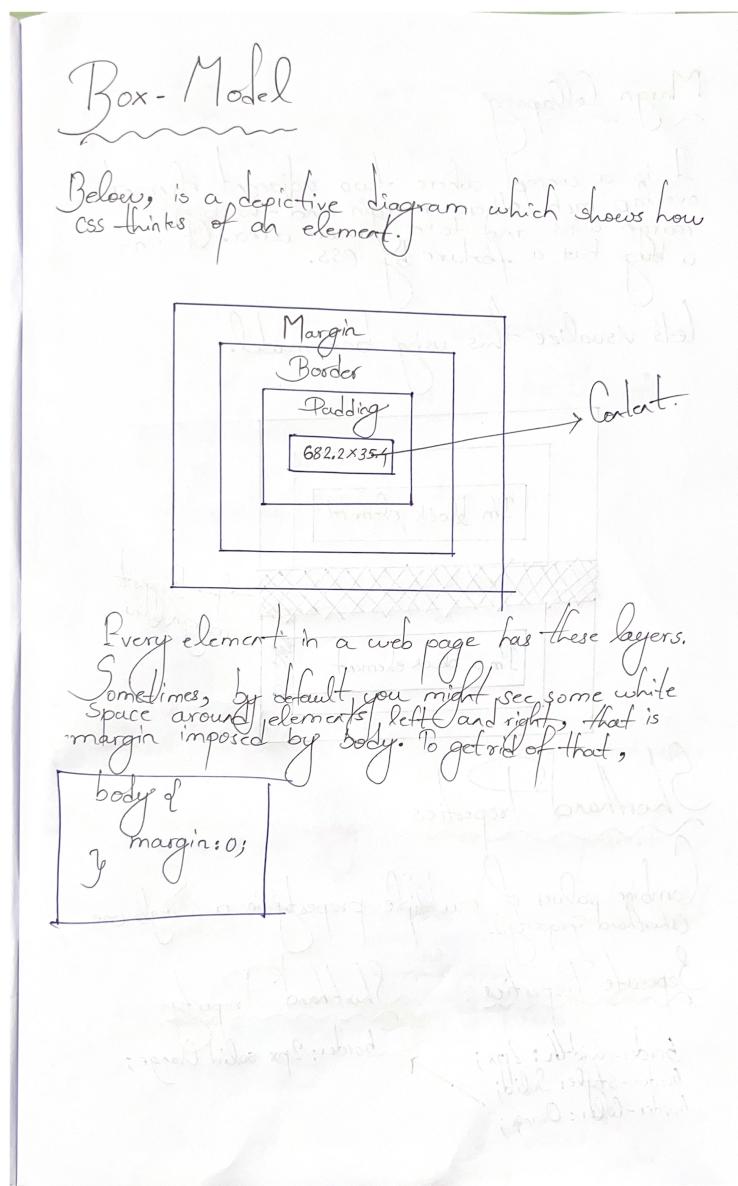


# Box-Model

The Box Model in CSS is a concept and layers of properties added to every element present in HTML. In this module we dive deeper into it,



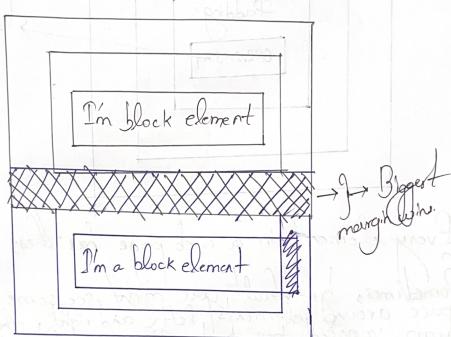
# Margin Collapsing

It's a Concept where two adjacent elements overlap each others margin and the one with the big margin wins and take more space. Read for more info ...

## Margin Collapsing

It is a concept where two adjacent element overlap each others margin and the bigger margin wins and takes more area. It isn't a bug but a feature by CSS.

Let's visualize this using box-model.



## Shorthand Properties

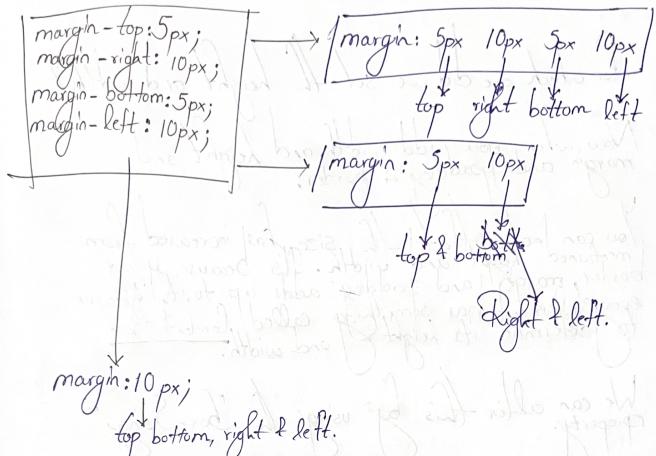
Combine values of multiple properties in a Single one.  
(Shorthand property).

### Separate Properties

border-width: 2px;  
border-style: Solid;  
border-color: Orange;

### Shorthand Property

border: 2px solid Orange;

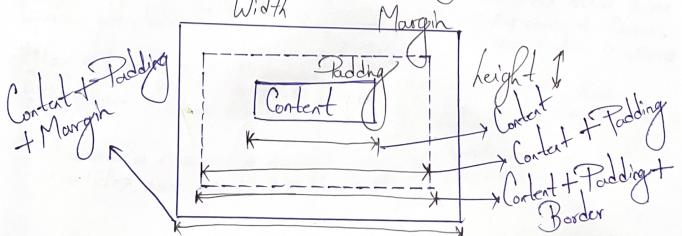


## Height & Width

Width is a property in which you can set value in percentages relative to the page width or exact value in pixels (px).

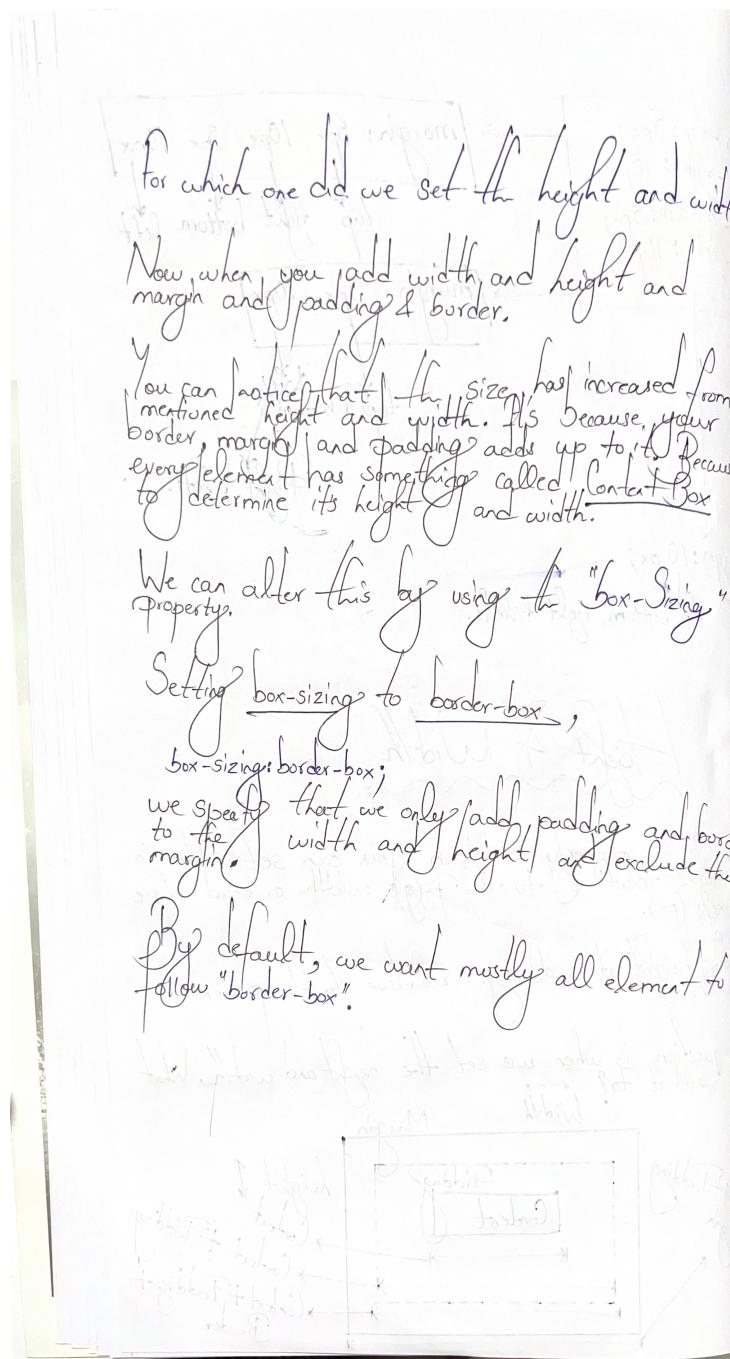
But for height it is always relative to its parent.

Now, question is when we set the height and width. What did we add it to?



# Height & Width

Now, you already have seen some of info about this topic in the Previous image. It's more about Height and Width and three behaviour at different times. Let's see it in more detail...



# Display Property

Various times we have to work with block & inline elements and toggle their type, for different behaviours. To do the same and understand more deeply, let's look into this.

## Display Property

By default when you put out some li items, they are block-level elements i.e. they take up maximum width and height possible. Usually, placed one after other.

Now, we can alter this, using "display" property and set inline-block and vice versa.

li { display: block } → It is default

li { display: inline } → It is used to convert block level elements to inline.

Now, for a navbar setup we need all in one row, that we can use,

li { display: inline-block } → It converts the block level elements into inline fashion but keeps their original width, height etc. property.

li { display: none } → It hides the element from screen but still remains in DOM.

### Block level

i) Rendered as block and take up all horizontal space possible.

ii) Can set margin-top and margin-bottom with respect to two different lines.

iii) Examples are: div, section, article, nav, h1, h2 & p.

Block elements

- i) Takes up only as much space necessary to fit content. Hence, two inline elements fit in same line.
- ii) It uses box-model but margin-top and margin-bottom has no effect on it. Padding top and bottom also has different effect.
- iii) Examples are: with a span, img.

\*\*\* Setting width and height in inline elements also have no effect. It is auto to take as much space as required by the content.

\*\*\*\*

While working with inline-block sometimes you will notice that the element sitting next to one element as inline-block doesn't come to same line but next line.

That is because, it's weird but it is actually your extra line space/indentation in your HTML file.

A workaround for this is either you remove the indentation or subtract the required amount from the 100% width of the container or element using calc() function.

Now, while working with a page and inline-block elements we often face two problems. Removing default link styles and positioning.

text-decoration: none;  
vertical-align: middle;

Remove link style applied by browser.

Places the inline-block container in the middle of its parent.

In the middle of

# PseudoClasses, Elements & Grouping Rules

Let's look at this in a bit uniform and brief way,

