**Box model**

A picture containing text, screenshot, rectangle, font

Description automatically generatedbox size

**width : width + padding left + padding right +border left+border right**

**height : height + padding top + padding bottom + border top +border bottom should remember always**

**Note : when no box-sizing property is set the default width and hight of element is calculated using the above formula and the width/height of element will be big than the original width/height since the padding or border set to element is added as extra with given width .. eg: 100px – original width +10px padding(right)+10px padding(left).**

When box-sizing : border-box then the width of element will be 100px even if we give 10px padding

\*;\*:before;\*:after {box-sizing:inherit} – this means the box-sizing given for elements will be inherited to all its associated elements. i.e if box-sizing:border-box for a element then it will be inherited to its child element also i.e child element also will be box-sizing:border-box

\*;\*:before;\*:after {box-sizing:border-box} – it means all main elements where box-sizing is not defined will have box-sizing as border-box and its child also will have box-sizing as border-box and if there is div with box-sizing : content-box and the child inside it will still have border-box since we used \*:after – which represent pseudo element

Note : div:before – this

box -sizing 🡺 border-box – the padding and border values move inside the element i.e the padding and border values are not considered for a element size . width 🡪 actual width of element box size

height -> actual height of element box size.

Box-sizing property in css control how box model should be handled.How the size of box should be determined.

Box-sizing = border-box – this is universal box sizing and it is better choice

Eg: .my{

Width : 200px

Border : 10px

Padding : 20px

} --width of box will be 200px

If box-sizing = content-box then the width of box in above example will be 200 of content+40(of padding)+20 px of border total width : 260px

**Inherit property in css:**

Using inherit will inherit the property from parent element

Eg:

<div class=”ele” style=”color:Red”>

<p>fff</p></div>

.ele P{

Color:inherit -- > this will inherit red color to p .. Note : inline style will override the inherit property i.e when inline property given for para it will override the inherit property

}

P{

Color:green,

}

Eg:2

<div style=”color:green”>teststt<p>ggg</p></div>

div p --this is inherit div color to p element inside div

{

Color:inherit;

}

**Initial property**

It is used to define the default initial value of any property

**Css selector:**

**Pseudo Elements vs Pseudo Classes**

::before – these are psedo elements that select/create some element

Pseudo classes – select element In certain condition

Pseudo class selectors are CSS selectors with a colon preceding them. You are probably very familiar with a few of them. Like hover

Types of selector:

1.class ---.cls{}

2.id--- #my{}

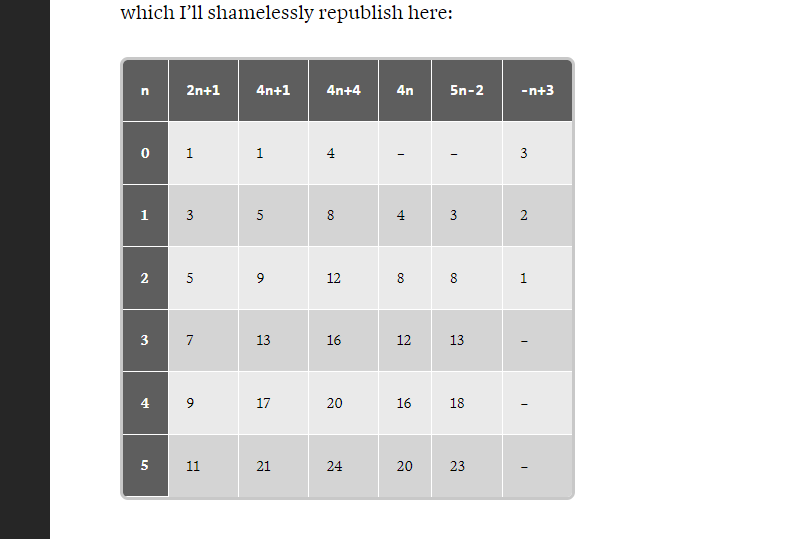
3.attribute selector -- input[type=text] -based on attribute given in html tag – same specificity of class 0010

4.tag slector – p{}

5.posititonal selector - :nth-of-child , :nth-of-type(3)

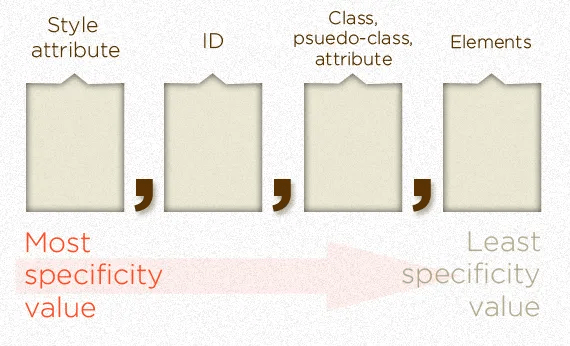
Nth-of-child(odd) – select odd element inside child

https://www.sitepoint.com/getting-to-know-css3-selectors-structural-pseudo-classes/



6.pesudo selector - a:hover , a:link a:visited https://css-tricks.com/pseudo-class-selectors/

**Specificity in css:**



Based on specificity the style will be applied to element

ID have more specificity than class selector

* Universal selector \* don’t have specificity
* Peuso element get 0001 --- eg: ::before get low specificity
* Pseudo-class :not – don’t have no specificity.
* !important value appended to any css property wins always. They take high priority than all other i.e 1,0,0,0. The only way an !important value can be overridden is with another !important rule declared later in the CSS and with equal or great specificity value otherwise

Pesudo classes:

:root -- root element i.e html tag it target

:only-child -- used to target child if it is only child of given element

:empty -- it targets element with empty value eg: <p></p>

:nth-child(n) -- target child based on n position given within its parent element

:nth-last-child(n) - order determined from last child element

:first-of-type -first element of specific type eg: p:first-of-type – 1st para

:last-of-type -target last element within given parent

:only-of-type

:nth-of-type(n) – target specific element according to its position relative to parent element.

:nth-last-of-type(n – relative position is taken from last element

:first-child - targets first child of parent

:last-child -target last child of parent

### **Combinators**

p>h1 – this is combinator

p strong – strong element that is child of p

next silbing cominator

--used to select immediate sibling followed by element using + symbol

H1+p – select first p element which is child of h1

.top \* + \* --this means select all element which is child of top

**Relative units in CSS:**

Eg: REM, EM, VW, VH

Specificity in css:

The selector with high priority overrides the other styles given for particular tag

**1.Universal selector:**

\*{} --this get 0 point of specificity

2.element or pseudo-element-selector(::after,::before) takes **1 point** of specificity

3.class,psudo-class-selector(:hover,:focus),attribute([href=”\*”]) – **10 points**

**Note:**

Div:not(.my-class) –adds nothing to specificity calculation but the element given in it have specificity

.my-class –10 points , div(element selector) -1 point

4. Id selector – **100 points**

**5.**inline style attribute **– 1000 points**

6.!important rule – **10000 points**