Thursday, 15 December 2022

CSS

**Combinators**

* The CSS elements has the prior level of inline>id>class>element
* If we want to apply a style to a specific element that comes inside or after a particular element we use combinators.

- Specificity is about resolving conflict when multiple rules target the same element.

* For example we have a h1 tag that comes inside a particular div and we have another h1 tag which is outside the div. If we want to style the h1 which is inside the particular div with the id we can add the id of the div along with the h1 in the css file so that the style will be only applied to the specific element.

Ie, #product-title h1{

color:green;

}

* The more specificity you mention the more priority it will have.
* If you use inherit keyword as the value to any property it will inherit that attribute from the parent or the tag which wraps that element.
* Combinators let you add more detail to your selector
* there are 4 important combinators

1. Adjacent sibling combinator

Added by using + symbol along with the selector. Eg:

h2 + p{

color:red;

}

In the above code the style applied to the h2 will be applied also for the paragraph tag which comes right after h2 tag. That means only direct siblings get the style. Another important thing is the elements should share the same parent.

1. General sibling combinator

uses ~ sign with the selectors. This is more flexible compared to the adjacent sibling selector.

h2 ~ p{

color:red;

}

in the above code the p tag need not to be a direct sibling of the h2 tag, in other words it not be placed right after the h2 tag. We only need to ensure that h2 tag and <p> tag are in the same level. Ie, it should have the same parent.

1. Child combinator

uses > sign with the selectors. It only applies style to the direct child element.

For example:- div > p{

color:red;

}

The paragraph tag that comes inside the div will get the the styling.

1. Descendant combinator

uses a white space with the selector. Here it doesn’t matter if it is a direct descendant or not. Any element that comes inside under any levels of nesting will have the style.

Eg:- div p{

color:red;

}

The second element must be descendant of the first element.

* Combinators used along with classes and id’s have great perfomance.
* The more specific you are the greater the performance.

**MDN CSS Reference we will get all the CSS properties and their values.**

**CSS Boxes**

* Every element in the html page is interpreted as a box in css.
* Padding is the internal space between the content and the border of the element
* Border surrounds the element and directly comes after the padding
* Margin is the distance between the element as the next sibling
* By default the browser sets some styles to the page. One such thing is margin for the body. Because of this there is a space between each element even after we set the margin. To avoid this we can set the margin to 0.
* If two block elements are placed next to each other. Each block has a margin. The block with the higher value of margin overlaps the margin of the block with smaller margin value. This is called margin collapsing. This is not a bug but to ensure that there is no huge space between two block elements.
* If you wish to take care of margin collapsing we can set the margin-top and Marin-bottom .

**Shorthand Properties**

* Combines values of multiple properties into a single property.
* Eg:-

you have

border-width: 2px;

border-style:dashed;

border-color:orange;

this can be shortened by using

border: 2px dashed orange;

The order doesn’t really matter as long as the value are not of the same kind.

* Another example of this is margin. It has a couple of shorthands.

margin-top:5px;

margin-right:10px;

margin-bottom:5px;

margin-left:10px;

we can write it as

margin:5px 10px 5px 10px;

top right bottom left

you can also set like

margin : 5px 10px;

this will set the same value ie, 5px for top and bottom and 10px for left and right.

Alternatively if we want to apply value for all sides we can use

margin:10px;

**Width and Height**

* By default width is set to 100% for block level elements.
* Width can be defined using percentages or absolute values in px
* In case of height element if we set the the height to 100% it will take the maximum height allowed by parent element. By doing this there won’t be any visible changes.So if we need to increase the size we need to change the height of the parent element.
* We can use absolute value in px to change the height. This will work as expected.
* The height and width will only be applicable to the content. If we add padding, border and margin the total height and width of the component will add up to the value of defined height and width.
* This is because of content box. Ie, every element has a default way for calculating width and height.
* If we define the box-sizing property we can adjust this behaviour.

box-sizing: content-box;

means that it will take the defined width and height for content only.

but if we define

box-sizing : border-box;

This will take the defined height and width to include the borders of the element.

eg;- if you define height : 528px;

border:5px

padding : 10px;

the height of the content will be 498px because (528 - (10) - (10) -(5)-(5)

will give 498. [2 times we are subtracting padding and border because of top and bottom]

* border-box is most commonly used box sizing. If we define this to the body due to inheritance some elements which have display:block property set as default will not be affected. So we use the universal selector \*{box-sizing:border-box;}. This will override the inheritance and target border padding and content size.

**Adding Header**

* we can use header to target a header, but there will be more than one header in a html page. So the best way to use it is to use a class or id.
* We use the display property to target the list defined in the header. The display property let’s us change the behaviour of an element from block to inline or to mix it as inline block or make to none which removes it from the visible document flow
* The inline-block is mix of both inline and block, it displays as inline and it allows us to set padding, border-top and border-bottom etc as block elements which are not possible on inline elements.
* We can apply this to list items. But remember that we cannot apply this to the entire list.
* **The display:none removes the element from the document flow which means other elements can take up its position. If we don’t want this we can use visibility:hidden . This will hide the element from the visibility but the place of the element will reserved so that other element cannot replace it.**
* **Also note that setting the width and height for an inline element is pointless since they are set automatically.**
* The calc() function is used to perform simple calculation in css.

Eg:- width : calc(100% - 49px)

note that the space between the value and the operator matters.

* To align an inline block to the right side first set the padding and margin to 0 then use text-align : right also make sure to set width to 100%
* If you want to make two block items in the same line give display:inline-block to both the items. If you want to move the 2nd item to the right side we can use the method in the above step but this will display the two elements in different lines. To avoid this we have to set the width 2nd item by subtracting the total width(100%) from the width of 1st item. Even though you do this it will not be aligned properly. This is because of the **white spaces in the code between 1st and 2nd item. To overcome this put both items in the same line one after the another. Another method is to assume a higher value for the 1st item and subtracting it.**

**Text Decoration and Header**

* To set the alignment of the text or a block vertically use the ‘vertical-align’ property.

eg: vertical-align:middle;

you must add the styling to the container.

-In some cases if the text-decoration property is not applied to a nested element we should use a combinator to target it. Or use a class for the inner element to which styling should be applied.

**Pseudo Classes**

* They are used to add effects to the elements. You can use various pseudo classes.

eg: a:hover{

color:white;

}

a:active{

color:white;

}

we can also use combinators to select particular elements.

* Note the :before the pseudo class name and there are no spaces.
* Pseudo classes let’s you define styles to a particular state of an element. We use :class name for that.
* Pseudo Elements are used to style a particular part of an element. To use this we use : : element name
* The ::after pseudo element is used to add extra content after the particular element.

Eg:- .main-nav\_\_item a::after{

content : “ (Link)”;

color:blue;

}

This will add a ‘ (Link)’ text after the anchor tag elements which will have a colour of blue.

* You can learn more about pseudo classes and elements in mdn reference
* We can also group multiple rules if they have same declaration using,

eg: - .main-nav\_\_item a:hover{

color:white;

}

.main-nav\_\_item a:active{

color:white;

}

can be combined into .main-nav\_\_item a:hover, .main-nav\_\_item a:active{

color:white;

}

* To add underline to a hyper link just set the border-bottom property

eg: border-bottom: 2.5px solid black;

instead of solid we can use dotted, or double

to get some space between the link text and the underline we can set the padding

like padding: 3px 0px;

this will set padding for top and bottom as 3px and left and right as 0px

* It is also possible to add more than one class to an element. For that just add a space between the two class names inside the double quotes.
* The border-radius property will round the corners. We use —cta with the class name for the elements which should stand out and perform some action.

**Adding Image as a background**

- this is done by using background:url(“path/file name.jpg/png”). The file can be from internet also Ie, we can specify https://url.