For just a normal tag directly tag name and you can go

For id #

For class .

Like

P {

Background-color : blue ;

}

#<the id> {

Background-color : blue ;

}

.<the class name>{

Background-color : blue ;

}

To cascade it

H1, h2 , h3 {

Background-color : blue ;

}

That means its applied to all the three

Body h1 {

Background-color : blue ;

}

It means that the h1 tag which is in the body tag

@ has been around since the days of @import in CSS1, although it's arguably becoming increasingly common in the recent @media (CSS2, CSS3) and @font-face (CSS3) constructs. The @ syntax itself, though, as I mentioned, is not new.

These are all known in CSS as **at-rules**. They're special instructions for the browser, not directly related to styling of (X)HTML/XML elements in Web documents using rules and properties, although they do play important roles in controlling how styles are applied.

Some code examples:

/\* Import another stylesheet from within a stylesheet \*/

@import url(style2.css);

/\* Apply this style only for printing \*/

@media print {

body {

color: #000;

background: #fff;

}

}

/\* Embed a custom web font \*/

@font-face {

font-family: 'DejaVu Sans';

src: local('DejaVu Sans Regular'), url(/fonts/DejaVuSans.ttf);

}

* [@font-face HYPERLINK "http://www.w3.org/TR/css3-fonts/" rules](http://www.w3.org/TR/css3-fonts/) define custom fonts for use in your designs that aren't always available on all computers, so a browser downloads a font from the server and sets text in that custom font as if the user's computer had the font.
* [@media HYPERLINK "http://www.w3.org/TR/css3-conditional/" rules](http://www.w3.org/TR/css3-conditional/), in conjunction with [media queries](http://www.w3.org/TR/css3-mediaqueries) (formerly only [media types](http://www.w3.org/TR/CSS21/media.html)), control which styles are applied and which aren't based on what media the page is being displayed in. In my code example, only when printing a document should all text be set in black against a white (the paper) background. You can use media queries to filter out print media, mobile devices and so on, and style pages differently for those.

In css3, attribute selector can also be done, like:-

P[name]{color:blue;}// that paragraph where attribute name is used

Or even more specific can be used:-

P[name=”FirstName”]{color:red;}// here every paragraph which has attribute name as FIrstName would get this css

In that also more specific can be done…using ‘^’ , ‘$’ or ‘\*’ operator… ^for the matching string in starting of the actual one… $ for the ending and \* for anywhere in the string in the actual one, example:-

P[name^=”Firstname”]{color:blue;} // ehre every paragraph whose attribute name starts with FirstName… e.x. :-

<p name=”FirstNameTest”></p> // this will be selected…likewise

We can do css of same class, i.e. pseudo class concept it is…:-

Something like this can be used to have any particular style according to n, e.x.:-

P:nth-child(3){color:blue;}// so here 3rd one would be in this style, not pretty much useful though

P:nth-child(odd)(color:blue;) // every odd element would be applied this style

P:nth-child(3n+1){color:blue;} // every element acc. To this formula would be styled

Similarly any formula can be applied…also the thing is to be written like this only… as In

<element name>:nth-child(<condition>)(<style>)

For negation pseudo class:-

<element tag may be applied>:not(<class name like .bukky>){color:blue;}

Okay for specifying that the element Is preceded by the same element we can do is:-

p.bukky + p{color:blue;}// this applies on all the p of class bukky which are immediately after a p

p.bukky ~ p{color:blue;} // this applies on all the p of class bukky which are after a p, but not necessarily immediately after it

parent child relationship , as :-

div > p {clor : blue;}// applie to those p who are child of div…\*note\* : this does not get applied to p which is child of some section or some other tag , and which is child of div… since then it wont be parent child relationship… it would be grandparent child relationship

div p {color: red;} // this will be applied to any p which is anywhere inside a div… not neccesarrily immediate

in the main wrapper class always mention width…1000px is the best option…

use float like float : left and also give width likewise..

clear : both; // for clearing all the float used above the tag…and i.e. back to normal

\*{..} // for overall website things such as margin padding

You should be able to utilize the asterisk and !important elements within CSS.

html \*

{

font-size: 1em !important;

color: #000 !important;

font-family: Arial !important;

}

The asterisk matches everything (you could probably get away without the html too).

The !important ensures that **nothing can override what you've set in this style** (unless it is also important). (this is to help with your requirement that it should "ignore inner formatting of text" - which I took to mean that other styles could not overwrite these)

---

notes taken in zeus:-

//we can use grid layout for doing things like in bootstrap.

//It is the CSS grid that is bundled with the bootstrap, that we are usually using.

display: grid;

//this is the start of making a css grid.

//fr is an attribute which can be used to describe 1 fragment, i.e. , 1fr is for 1 part of the available space.

//header should be used for semantic purposes for header.

//nav inside header should be used for navigation. again for semantic purposes

---

//divs and spans dont tell semantically what our code is all about.

//These are the vendor-prefixed properties offered by the relevant rendering engines (-webkit for Chrome, Safari; -moz for Firefox, -o for Opera, -ms for Internet Explorer)

//using javascript sniffing to make things responsive is the worst way, since it is very difficult to make difference between tablet and mobile.

// using css media queries , it is also not very good idea but it is better than javascript

//using media query is very easy, we just need to follow the syntax:-

@media not|only mediatype and (media feature) {

CSS-Code;

}

//where the last on in paranthesis is the condition for the activation of the code, i.e. just like an if statement.

//for wrapping the text around

word-wrap: break-word;

//also has many other options

//if we are applying percentage in margins or padding or something in vertical then it will wary according to the width of the container.

//box-sizzing is a very good attribute, if set to border-box then all the padding and margins and borders to that element would be aplied into that same size, i.e. If width is set as 100px, then padding, margin , boder everything is taken from that 100px

//There are various attributes that can be used for sizing , it can be set to fit-content, inherit, the parent element can be set to a size accroding to percentage, set its overflow : hidden or just in that direction, and in the child element set padding in that dimension as 5% or 100% or something or infact even if we dont set padding that dimension will be of that size

//sprites are a collection of small images stored in one so as to reduce the number of http requests and therefore also to reduce the loading time, it is the same as we had small smaal images in our KIRC project slider for arrows etc.

//reflows and repaints are expensive, we should try to reduce it as much as possible.

//very good suggestions to reduce reflows and repaints

//hide the element with display: none (1 reflow, repaint), add 100 changes, restore the display (another reflow, repaint). This way you trade 2 reflows for potentially a hundred

//clone the node you're about to update, work on the copy, then swap the original with the updated clone

//reducing the number of time using the actual variables, especially when in big loops, instead use local var, store it there and boom , reflows and repaints are reduced.

//Width can also be set as min/max – content.

//Also we can use fill.

//we could use position: sticky for footers. and not need to play around with js unneccessary .

//:nth-child(n+3):nth-child(-n+7)

//css range selector

if we apply id and css to a same element and apply different properties then id's changes would be reflected