* pseudo-elements
* **: first-letter, :: first-letter** these two are used to add a special style to the first letter of a text.
* **: first-line, :: first-line** these two are used to add a special style to the first line of a text.

**Note\*** The single colon was used when it was still CSS1 and CSS2, then replaced the single colon to double colon when it is already CSS3.

**CSS Rule Precedence**

* an HTML element may be the subject of the selectors of multiple style rules.
* such rules target different properties, their effects cascades.
* the styles involve the same property, hey conflict, and must be resolved such that only one style is applied.
* resolution:
  + by **Origin and Importance**
  + by **Specificity**
  + by Order

*Origin and Importance*

* the origin of a declaration is based on where it comes from and its importance is whenever it is declared important. The precedence of the various origin is in descending order:

1. Transition Declaration
2. Important user agent declarations
3. Important user declarations
4. Import override declarations
5. Import author declaration
6. Animation declaration
7. Normal override declaration
8. Normal author declaration
9. Normal user declaration
10. Normal user agent declaration

Declarations from origins earlier in this list win over declarations from later origins.

*Specificity* (Calculating a selector’s specificity)

A selectors specificity is calculated as follows:

* + count the number of ID selectors in the selector (=a)
  + count the number of class selectors, attribute selector, pseudo-classes(=b)
  + count the number of type selectors and pseudo-elements in the selector(=c)
  + ignore the universal selector

**CSS Declarations**

* properties
* shorthand properties
* allows authors to specify the values of several properties with a single property.
* Examples: background, font, margin, padding, border, border-left, border-width, etc.
* vendor specific extensions (also known as vendor prefixes)
* used by browser vendors as a prefix for the names of experimental or non-standard CSS properties; lately, vendors are moving away from vendor prefixes in favor of user-controlled flags or preferences
* Examples: -webkit-, -moz-, -o-, -ms-, etc.
* custom properties, also known as, CSS variable (experimental)
* property names prefixed with --, representing a value that can be reused throughout the document using the var() function.
* values
* *value processing*
* declared, cascaded, specified, computed, used, actual values
* value types
* keywords
* CSS-wide keywords
* Numbers
* integers or reals in (scientific) decimal notation
* Dimensions
* length, angle, duration (or time), frequency, resolution
* *length units*:
* font-relative: em, ex, ch, rem
* viewport-percentage: vw, vh, vmin, vmax
* absolute: cm, mm, in, pt, pc, px
* *angle units*:
* deg, grad, rad, turn
* used in some gradient and transform functions
* duration (or time) units:
* s, ms
* used in animation, transition, and related properties
* frequency units:
* Hz, kHz
* initially introduced in CSS2 for the (obsoleted) *aural* media types; reintroduced in CSS3, but is currently unused
* resolution units:
* dpi, dpcm, dppx
* used in media queries
* percentages
* number with a % suffix
* calculated as a percentage of some value (usually taken from the parent element).
* *URLs and URIs*
* url() function with an absolute or relative (to the stylesheet) URL parameter.
* demotes a pointer to a resource, such as an image or a font.
* Colors
* colors keyword
* RGB hexadecimal notation
* Examples: #ff0000, #ff0000ff, #f00, #f00f
* RGB Function
* Examples: rgb (255,0,0), rgb (100%, 0%, 0%), rgba (255, 0, 0, 1)
* HSL Function
* Examples: hsl(0, 100%, 50%), hsla (0, 100%, 50%, 1)
* currentcolor, transparent
* Strings
  + - Delimited by single quotes (‘) or double quotes (“)
* functions
  + - Used as a value for various CSS properties
* Examples: attr(), var(), rgb(), hsl()
* miscellaneous types
  + - CSS properties that are highly experimental or don’t fit in any other categories.

**CSS Preprocessors, Frameworks and Polyfills**

* CSS Preprocessors
  + Generates CSS using a custom language syntax that typically includes features that don’t exist in pure.
  + Examples: SASS (Syntactically Awesome Stylesheets), LESS, Stylus, etc.
* CSS Frameworks
  + provides predefined CSS design functionality that can be reused, extended or customized.
  + Examples: Bootstrap, Foundation, Materials, etc.
* Polyfills
* ways of detecting certain features that cannot be supported by a browser.
* provides predefined CSS.