# Chapter 1 - CSS fundamentals

## CSS Selectors

* **Universal Selector** - *\*, ns|\*, \*|\**

It matched all the elements of the document

* **Type Selector** – *elementname*

It matches all the elements of the document with the given node name.

* **Class Selector** - *.classname*

Selects all the elements of the document with the given class name.

* **ID Selector** - *#idname*

Selects all the elements of the document with the given ID name.

* **Attribute Selector** – [attr], [attr=value], [attr~=value], [attr|=value], [attr^=value], [attr$=value], [attr\*=value]

Example [type] – it will select all the elements with the “type” attribute

which is set to any value.

### Grouping Selectors

* “,” will select all the matching nodes.

Eg. – div, span will select all the div and span elements of the document.

* “ “ (space) descendants combinator selects nodes which are descendants of the first element.

Eg. – div span will select all span elements that are inside a div element.

* “>” child combinator selects the direct children of the first element.

Eg. – ul > li will select all the <li> elements that are direct children of <ul>

* “+” adjacent sibling combinator will select adjacent siblings of the first element.

Eg. – h2 + p will select all <p> elements which are adjacent to all h2.

### Pseudo

* Pseudo Classes – : pseudo allows the selection of elements based on a state which is not defined in DOM

Eg. - a:visited

* Pseudo Elements - :: pseudo represent entities that are not included in HTML.

Eg. – p::first-line

## Cascade

* A rule or ruleset in CSS means the entire block.

.example(selector) {

Color: red; (declaration)

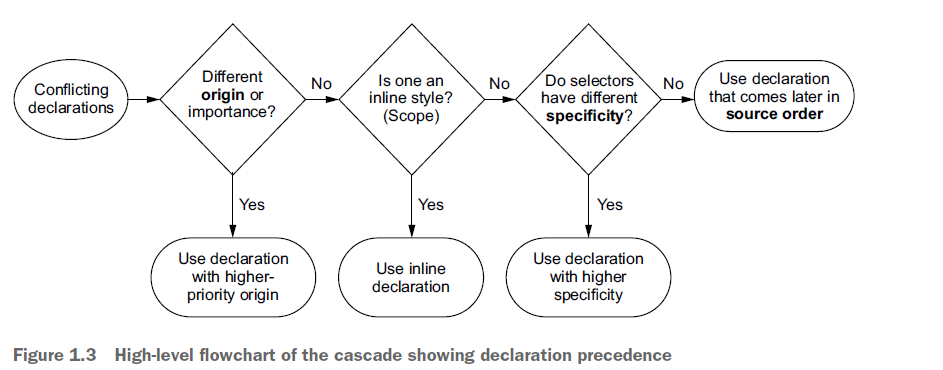
Font-family: sans-serif; (declaration) \*\*

}

\*\* - Yellow portion is called as declaration block

* Predicting how rule behaves requires an understanding of a cascade.
* When two or more rules have conflicting declarations, browser follows a set of rules to predict the result. These set of rules is called as Cascade.
* It tells us how conflicts are resolved.
* Cascade has the following 3 things to resolve a conflict.

1. **Stylesheet origin** – Where the styles come from. It is custom styles or the browser specific styles (user-agent stylesheet)
2. **Selector specificity** – Which selectors have precedence over which
3. **Source order** – Order in which the styles are declared in the stylesheet



### Understanding Stylesheet Origin

* Where the styles are coming from.
* User-agent styles are browser default styles for an element.
* Author styles are our styles.
* Author styles are given higher precedence over user-agent styles.
* Author important styles (!important) is given highest precedence.

1. Author important styles
2. Author styles
3. User-agent styles

### Understanding Specificity

* If the conflicting declarations cannot be resolved using origin, browser considers the specificity.

1. Styles applied inline
2. Styles applied using a selector

* Inline styles are applied using <style> attribute for the HTML element.
* Inline styles override any declarations present in your stylesheet or <style> tag.
* Inline styles are also called as “scoped” declarations.
* Selector specificity is determined by following rules:

1. If the selector has more IDs, it wins
2. If the results end in a tie, the selector with most classes wins
3. If the results end in a tie, the selector with most tags (type selector) wins

* Pseudo class selectors and attribute selectors have same precedence like a class selector.
* Universal selector and combinators (>,+,~) has no effect on specificity.
* Notation of specificity – (IDs, classes, tags)
* Three ways you can fix the specificity

1. Using !important
2. Increasing the specificity for the styles you want apply
3. Decreasing the specificity for the other selectors

* Try to avoid !important as much as possible.
* Try to keep the selector specificity as low as possible so that when you want to override, it becomes simple.

### Understanding Source Order

* When the selectors have same specificity, then it is resolved by the source order.
* The styles at the end wins in this case. OR when the stylesheet is linked at later parts of your document.
* When styling problem comes, consider these:

1. Think about which declarations will make a fix
2. Structure the selectors and choose the best which fits the needs

* Link styles should be in certain order

a:link

a:visited

a:hover

a:active

* The declaration that wins the cascade is called as cascaded value. It is the value given to the property applied to an element as a result of the cascade.

### Two rules of thumb

* Don’t use ID in your selector – Even 1 ID can increase the specificity of the selector
* Don’t use !important – This is even hard to override. Once you use this, you will have to add it every time you override.

## Inheritance

* Another way that and element get the styles is using the inheritance.
* We can apply font-family to <body>, so its descendants can inherit that property.

color, font, font-family, font-size, font-weight, font-variant,

font-style, line-height, letter-spacing, text-align, text-indent, text-transform,

white-space, and word-spacing.

* We can target specific elements of the page so that only its descendants can inherit the property.
* We can view all the styles in the DevTools. The styles appear in the decreasing order of their specificity. That is, higher the styles higher is the specificity. Below all these are inherited styles. It shows from where the style is getting inherited.
* There are two special values.

1. Inherit – If you want to inherit the style from its parent by overriding a cascaded value, you can use inherit keyword.
2. Initial – Sometimes you want some styles to undo. In those cases you can use initial. Its like a hard-reset for that property. Initial value is for the property, not element. It resets the value of that property to its initial or default value. Its better to use initial rather than auto, because there are some properties which do not have auto value.

## Shorthand properties

* Shorthand properties are those which allows you set the values of several other properties at one time.
* Shorthand properties silently override values. If all the properties are not specified in shorthand, then the omitted properties are set to their initial values.
* When it comes to order, shorthand properties does not have any pre set order. Because browser knows the value corresponds to which property.
* But for certain properties like margin and padding, the order of values matter. In the properties where we are mentioning the values for all the 4 sides of an element, order matters. Its is TOP, RIGHT, BOTTOM and LEFT.
* You can use truncated notation for shorthand properties. When truncated, the value which is not mentioned takes the value from its opposite side.
* It works in VERTICAL – HORIZONTAL manner.
* There are elements like background-position, box-shadow, text-shadow which uses two values only. Those work in HORIZONTAL – VERTICAL manner. Because they follow a cartesian grid – (x,y).