Cascading Style Sheet

Cascading Style Sheet

“Cascading" *means* that styles can fall (or *cascade*) from one style sheet to another, enabling multiple style sheets to be used on one HTML document.

Here's a quick recap of the browser's steps:

1. Process HTML markup and build the DOM tree.
2. Process CSS markup and build the CSSOM tree.
3. Combine the DOM and CSSOM into a render tree.
4. Run layout on the render tree to compute geometry of each node.
5. Paint the individual nodes to the screen.

History of CSS

CSS was first proposed by **Hakon Wium Lie** on October 10, 1994. At the time, Lie was working with Tim Berners-Lee (father of HTML) at CERN. The European Organization for Nuclear Research is known as CERN

IN 1990 Tim Berners-Lee wrote his NeXT browser/editor in such a way that he could determine the style with a simple style sheet. However, he didn't publish the syntax for the style sheets, considering it a matter for each browser to decide how to best display pages to its users.

In 1992, Pei Wei developed a browser called Viola, which had its own style sheet language heir users fewer and fewer options to influence the style.

In 1993, NCSA Mosaic, the browser that made the Web popular, came out. Stylewise, however, it was a backward step because it only allowed its users to change certain colors and fonts.

On October 13, 1994, Marc Andreessen announced to *www-talk* that the first beta release of Mozilla.

Three days before Netscape announced the availability of its new browser, Håkon published the first draft of the *Cascading HTML Style Sheets* proposal. Behind the scenes, Dave Raggett (the main architect of HTML 3.0) had encouraged the release of the draft to go out before the upcoming Mosaic and the Web conference in Chicago.

Dave had realized that HTML would and should never turn into a page-description language and that a more purpose-built mechanism was needed to satisfy requirements from authors.

Among the people who responded to the first draft of CSS was Bert Bos. At that time, he was building Argo, a highly customizable browser with style sheets, and he decided to join forces with Håkon.

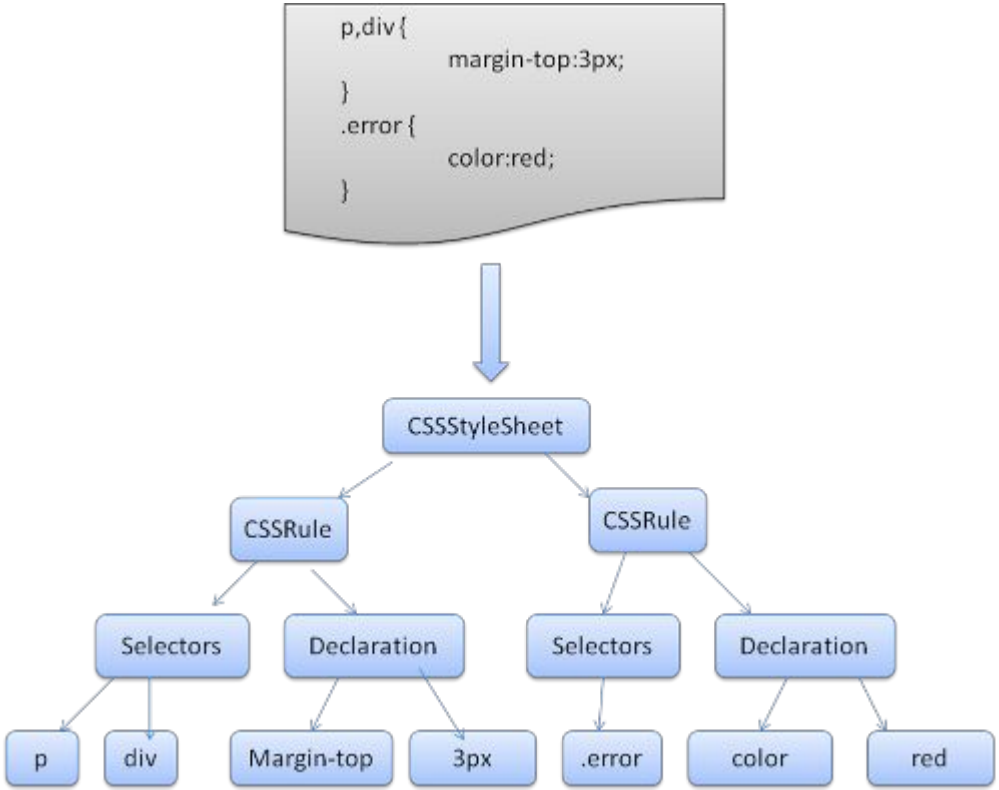
After w3c become active in 1995. Bert presented the support for style sheets in Argo, and Håkon showed a version of the Arena browser that had been modified to support CSS

www-style mailing list was created in May 1995, and the discussions there have often influenced the development of the CSS specifications.

it was decided to hold the workshop in Paris,Among the participants was Thomas Reardon of Microsoft, who pledged support for CSS in upcoming versions of Internet Explorer. After Microsoft signaled that it was adding CSS support in its browser, it was also important to get Netscape on board.

In February 1997, CSS got its own working group inside W3C and the new group set out to work on the features which CSS1 didn't address.

CSS level 2 became a Recommendation in May 1998. Since then, the group has worked in parallel on new CSS modules and errata for CSS 2. The W3C working group has members (around 15 in 1999, around 115 in 2016) who are delegated by the companies and organizations that are members of W3C



CSSDOM Structure

Web Engine Browser

Blink
Google Chrome and Chromium, plus many other browsers including Opera and Vivaldi

EdgeHTML
Microsoft Edge browser and Universal Windows Platform apps

Gecko
Firefox browser and Thunderbird email client, plus forks like SeaMonkey and Waterfox

WebKit
Safari browser, Adobe AIR apps, and other browser like Maxthon

KHTML - Discontinued
Konqueror browser

Presto - Discontinued
formerly in the Opera browser

Trident - Discontinued
Internet Explorer browser and Microsoft Outlook email client

Precedence

- Inline
- Internal
- External

Rules

1. inline css (html style attribute) overrides css rules in style tag and css file
2. a more specific selector takes precedence over a less specific one
3. rules that appear later in the code override earlier rules if both have the same specificity.
4. A css rule with `!important` always takes precedence

Specificity

What Is Specificity?

If two CSS selectors apply to the same element, the one with higher specificity wins.

Specificity for single selectors from highest to lowest:

- ids (example: `#main` selects `<div id="main">`)
- classes (ex.: `.myclass`), attribute selectors (ex.: `[href=^https:]`) and pseudo-classes (ex.: `:hover`)
- elements (ex.: `div`) and pseudo-elements (ex.: `::before`)

To compare the specificity of two combined selectors, compare the number of occurrences of single selectors of each of the specificity groups above.

Example: compare `#nav ul li a:hover` to `#nav ul li.active a::after`

- count the number of id selectors: there is one for each (`#nav`)
- count the number of class selectors: there is one for each (`:hover` and `.active`)
- count the number of element selectors: there are 3 (`ul li a`) for the first and 4 for the second (`ul li a ::after`), thus the second combined selector is more specific

CSS Colors

- Keyword - color:red;
- RGB - Absolute - rgb(RR,GG,BB)
- RGB - Percentage - rgb(R%,G%,B%)
- RGBA - rgba(RR,GG,BB,Alpha) ** Alpha takes values from 0 -1
- HEX Code - rgb(R%,G%,B%)
- HEX Short Code - #RRGGBB
- HSL
- HSLA

hsl(*hue, saturation, lightness*)

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

CSS Background

- The **background-color** property is used to set the background color of an element.

  background-color: green;

- The **background-image** property is used to set the background image of an element.

  background-image: url("paper.gif");

- The **background-repeat** property is used to control the repetition of an image in the background.

  background-repeat: repeat-x; -- horizontal

  background-repeat: repeat-y; -- vertical

  background-repeat: no-repeat; - Specified Only Once

- The **background-attachment** property is used to control the scrolling of an image in the background.

  background-attachment: fixed || scroll;

- The **background-position** property is used to control the position of an image in the background.

  background-position: right top;

Shorthand

background: #ffffff url("img_tree.png") no-repeat right top;

CSS Borders

Border Style

The border-style property specifies what kind of border to display.
The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

- **border-width** property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

border-width: 2px 10px 4px 20px;

- **border-color** property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- Hex - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- transparent

Border - Individual Sides

- From the examples above you have seen that it is possible to specify a different border for each side.
- In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

Shorthand :

Color

```
p.example1{
        border:1px solid;
        border-bottom-color:#009900; /* Green */
        border-top-color:#FF0000;    /* Red */
        border-left-color:#330000;   /* Black */
        border-right-color:#0000CC;  /* Blue */
      }
```

Left Border

```
p {
   border-left: 6px solid red;
         background-color: lightgrey;
}
```

- **border-radius** property is used to add rounded borders to an element

```
p {
         border: 2px solid red;
          border-radius: 5px;
}
```

Shorthand

```
p {
   border: 5px solid red;
}
```

Height & Width

The height and width properties are used to set the height and width of an element.

The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block.

```
div {
        height: 200px;
       width: 50%;
       background-color: powderblue;
}
```

CSS margin

margin properties are used to create space around elements, outside of any defined borders.

CSS has properties for specifying the margin for each side of an element:

- margin-top
- margin-right
- margin-bottom
- Margin-left

Example :

```
p {
   margin: 25px 50px 75px 100px;
}
```

CSS padding

CSS padding properties are used to generate space around an element's content, inside of any defined borders.

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- Padding-left

```
div {
 padding: 25px 50px 75px 100px;
}
```

CSS Text

- The **color** property is used to set the color of a text.

```
h1 {
        color: green;
}
```

- The **direction** property is used to set the text direction.

```
<p style="direction:rtl;">
```

- The **letter-spacing** property is used to add or subtract space between the letters that make up a word.

```
h1 {
        letter-spacing: 3px;
}
```

- The **word-spacing** property is used to add or subtract space between the words of a sentence.

- The **text-indent** property is used to indent the text of a paragraph.

```
p {
        text-indent: 50px;
}
```

- The **text-align** property is used to align the text of a document. right , left, center, justify

- The **text-decoration** property is used to underline, overline, and strikethrough text.

```
h2 {
        text-decoration: line-through;
}
```

- The line-height property is used to specify the space between lines:

```
line-height: 0.8;
```

- The **text-transform** property is used to capitalize text or convert text to uppercase or lowercase letters.

- The **text-shadow** property is used to set the text shadow around a text.

The text-shadow property adds shadow to text.

The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red):

Example

```
h1 {
        text-shadow: 3px 2px red;

}
```

CSS Font

Font Family

The font family of a text is set with the font-family property.
The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.

```
p {
        font-family: "Times New Roman", Times, serif;
}
```

- **font-style** property is mostly used to specify italic text.

This property has three values:

- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Set Font Size With Em

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.
The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

Example

```
h1 {
        font-size: 2.5em; /* 40px/16=2.5em */

}
```

The font-weight property specifies the weight of a font: normal, bold

Example

```
p.normal {
        font-weight: normal;

}
```

Responsive Font Size

The text size can be set with a vw unit, which means the "viewport width".
That way the text size will follow the size of the browser window:

```
<h1 style="font-size:10vw">Hello World</h1>
```

CSS Display

display property specifies if/how an element is displayed.

A **block-level** element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.

An **inline** element does not start on a new line and only takes up as much width as necessary.

CSS Positions

The position property specifies the type of positioning method used for an element.

There are five different position values:

- Static - is not positioned in any special way; it is always positioned according to the normal flow of the page
- Relative - is positioned relative to its normal position
- Fixed - is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.
- Absolute - absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
  However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.
- Sticky - is positioned based on the user's scroll position.

The z-index property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position:absolute, position:relative, or position:fixed).

cursor

.alias {cursor: alias;}

.all-scroll {cursor: all-scroll;}

.auto {cursor: auto;}

.cell {cursor: cell;}

.context-menu {cursor: context-menu;}

.col-resize {cursor: col-resize;}

.copy {cursor: copy;}

.crosshair {cursor: crosshair;}

.

default {cursor: default;}

.e-resize {cursor: e-resize;}

.ew-resize {cursor: ew-resize;}

.grab {cursor: grab;}

.grabbing {cursor: grabbing;}

.help {cursor: help;}

.move {cursor: move;}

.n-resize {cursor: n-resize;}

.

.not-allowed {cursor: not-allowed;}

.pointer {cursor: pointer;}

.progress {cursor: progress;}

.text {cursor: text;}

.wait {cursor: wait;}

.zoom-in {cursor: zoom-in;}

.zoom-out {cursor: zoom-out;}

Media Types

The `@media` rule, introduced in CSS2, made it possible to define different style rules for different media types. Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.

```
@media screen and (min-width: 480px) {

    body {

        background-color: lightgreen;

    }

}
```

CSS Column

column-count

Specifies the number of columns an element should be divided into

column-gap

Specifies the gap between the columns

column-rule-color

Specifies the style of the rule between columns

column-rule-width

Specifies the width of the rule between columns

Column-width

Specifies a suggested, optimal width for the columns

The vendor-prefixed properties are used to implement new, or proprietary CSS features, prior to final clarification/definition by the W3C.

This allows properties to be set specific to each individual browser/rendering engine in order for inconsistencies between implementations to be safely accounted for. The prefixes will, over time, be removed once the feature reaches the final version.

It's usually considered good practice to specify the vendor-prefixed version first and then the non-prefixed version.

```
div {
      -webkit-columns: 100px 3; /* Chrome, Safari, Opera */
      -moz-columns: 100px 3; /* Firefox */
      columns: 100px 3;
}
```


THANK YOU