CSS Notes By Geekster

Geekster

www.geekster.in

CSS NOTES

What is CSS and Purpose

CSS (Cascading Style Sheets) is **used to style and layout web pages** — for example, to alter the font, color, size, and spacing of your content, split it into multiple columns, or add animations and other decorative features.

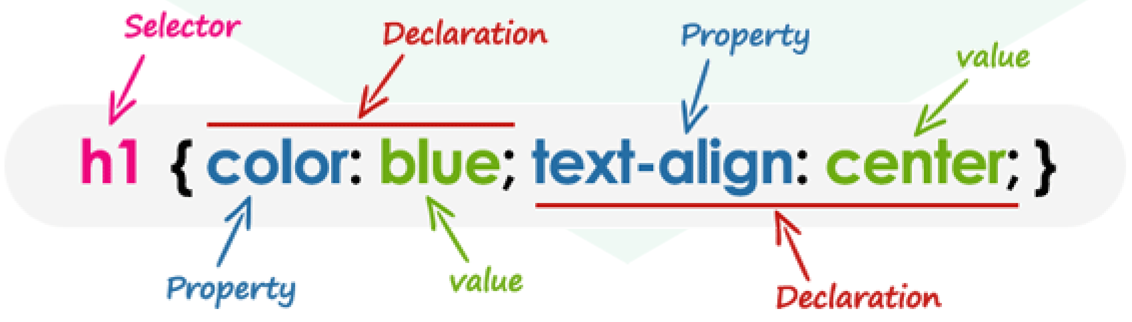
Basic CSS

- **The Cascade**: Process of combining different stylesheets and resolving between different CSS rules and declarations, when more than one rule applies to a certain element
- Author (me), User (client), Browser(user agent) styles
- Importance > Specificity > Source Order
- **Importance**
    - User !important
    - Author !important
    - Author declarations
    - User declarations
    - Default browser declarations
- **Specificity**
    - Inline styles
    - IDs
    - Classes, pseudo-classes, attribute
    - Elements, pseudo-elements Specificity ➡️ (Inline, IDs, Classes, Elements)
- **Source Order**
    - If above is equal, last rule is applied
- *Universal Selector* * has the specificity of (0,0,0,0)

- **Value Processing**:
    - Declared Value
    - Cascaded Value (winner among declared values)
    - Specified Value (default value if no cascade value)
    - Computed Value (relative units to absolute)
    - Used Value (final calc, based on layout, part of render)
    - Actual Value (rounding of Used Values)
- Relative Units (%, rem, em) to Absolute (px)
    - % fonts ➡️ based on parent's font-size
    - % lengths ➡️ based on parent's **width**
    - em fonts ➡️ based on parent's font-size
    - em lengths ➡️ based on current elements font-size
    - rem ➡️ based on root element font-size
    - vh ➡️ viewport height
    - vw ➡️ viewport width

CSS syntax

- Learning CSS / CSS Syntax is remarkably very simple and easy
- A CSS rule consists of a set of styles/rules that are interpreted by the browser and then applied to the corresponding elements in the document



Syntax:

```
selector {
    property: value;     /* declaration/rule 1 */
    property: value;     /* declaration/rule 2 */}
```

Basic elements of CSS syntax:

- A CSS style rule is made of the following parts:

Item	Description
Selector	Target (it may be an HTML Element/Tag/Id/Class)
Declaration/rule	Definition part in curly brace { property: value; }
Property	skin/style attribute to change (font, size, color, border)
Value	value to apply to CSS properties (Arial font, 10px size, red color)

Note: Writing selector is just half part of CSS rule, the rest half belongs to property: value I want to set.

Basic Selector types

- Selector helps to target an elements on the page
- The selector allows us to tell the browser which element/tag on the page we want to style
- CSS selectors are used to target/find/select HTML elements based on element/tag name, id, class, attribute, and more
- Selector may target: Single element, group of elements, .class, #id, any specific conditional element
- Selectors are used to declaring which of the markup elements a style applies to, Selector indicates the HTML element you want to style

Element/Tag/Type Selector

- The element selector target/find/selects the HTML element/tag by name
- Global selectors based on any HTML element/tag name

- This is the simplest/basic way to target all elements of a given element/tag/type

Syntax:

```
Element/Tag {
    property: value;    /* declaration/rule 1 */}
```

Class Selector

- The class selector selects HTML elements with a specific class attribute
- It is used with a period character . (full stop symbol) followed by the class name
- Attribute selector based on class attributes applied to HTML element/tag
- A generic set of rules applied to the similar-looking item (belongs to same .class)
- .class is a very important attribute while dealing with HTML+CSS as similar looking/output based things on a web page usually belongs to one class
- .class selector is a more precise way to target an element/markup as compared to ab element/tag/type selector

Note: .class and #id attributes/selectors not only allows to style content but primarily exists to give more meaning/semantics to content.

Syntax:

```
className {
    property: value;    /* declaration/rule 1 */
}
<h1 class="className">Some heading content </h1>
```

Id Selector

- The id selector selects the id attribute of an HTML element to select a specific element
- It is used with a hash/pound character # sign followed by the id name
- Attribute selector based on specific id attributes applied to HTML element/tag

- An id is always single/unique within the page so it is chosen to select a single, unique element in a web page
- #id is a very important attribute while dealing with HTML+CSS to make and target single unique thing in document

Note: #id and .class attributes/selectors not only allows to style content but primarily exists to give more meaning/semantics to content.

Syntax:

```
<h1 id="idName">Some heading content </h1>
#idName {
    property: value;    /* declaration/rule 1 */
}
```

Universal Selector

- The universal selector selects all the elements on the page
- Rather than selecting elements of a specific type, the universal selector simply matches the name of any element type
- The universal selector is used as a wildcard * character

Syntax:

```
* {
    property: value;    /* declaration/rule 1 */
}
```

Group/Grouping Selector

- Group/Grouping selector allows you to group different types of selectors together that share similar formatting
- Grouping selector is used to minimize the code, its highly efficient selector
- It also prevents you from repeating the same style rules over and over again
- You can apply a style to many selectors, Just separate the selectors with a comma (,)
- Commas , are used to separate each selector in grouping

Syntax:

```
element,.class,#id, element selector{
    property: value;    /* declaration/rule 1 */
}
```

Conditional/Descendant Selector (parent-child)

- Highly specific selector that target elements based on their location within other elements
- You can use this selector when you need to select an element that is the descendant/inside of another element
- You want to apply a style rule to a particular element only when it lies inside a particular element
- White space is used to denote a hierarchy of elements (parent nested child)
- Conditional/Descendent selector is used to minimize the code, its highly efficient selector

Syntax:

```
parent child selector{
    property: value;    /* declaration/rule 1 */
}
```

Pseudo-class Selector

We used many selectors like element, class, id, universal etc. which are fully dependent on DOM structural representation (Document Object Model). But special states like :hover, :active, :visited, :before :after are styled with Pseudo-class Selector.

- pseudo means not actually but having the appearance of; pretended; false or spurious;
- Pseudo-class selectors are CSS selectors with a colon preceding them :
- A CSS pseudo-class is a keyword added to a selector that specifies a special state of the selected element(s) like hover, active, focus, visited, link

- Pseudo-classes let you apply a style to an element not only in relation to the content of the document tree
- A pseudo-class is basically a special state or specific characteristic of an element that can be targeted with CSS. A few common pseudo-classes are :link, :visited, :hover, :active

Syntax:

```
element:link {
    property: value;    /* declaration/rule 1 */
}

element:visited {
    property: value;    /* declaration/rule 1 */
}

element:hover {
    property: value;    /* declaration/rule 1 */
}
```

```
element:active {
    property: value;    /* declaration/rule 1 */
}
```

CSS Properties

border-radius

```
-webkit-border-radius: 4px;
-moz-border-radius: 4px;
border-radius: 4px;
```

Allows rounded corners in elements. Can also be used to create circles.*Support: IE9+*

box-shadow

```
-webkit-box-shadow: 1px 1px 3px #292929;
-moz-box-shadow: 1px 1px 3px #292929;
box-shadow: 1px 1px 3px #292929;
```

The box-shadow property allows us to easily implement multiple drop shadows (outer or inner) on box elements, specifying values for color, size, blur and offset.

box-shadow accepts four parameters: x-offset, y-offset, blur, color of shadow.

- The first value defines the horizontal offset of the shadow, with a positive value offsetting the shadow to the right of the element, and a negative value to the left.
- The second value defines the vertical offset of the shadow, with a positive value offsetting the shadow from the bottom of the element, and a negative value from the top.
- If supplied, an optional third value defines the blur distance of the shadow. Only positive values are allowed, and the larger the value, the more the shadow's edge is blurred.
- An optional fourth value can be supplied to define the spread distance of the shadow. A positive value will cause the shadow shape to expand in all directions, while a negative value will cause the shadow shape to contract.

An optional 'inset' keyword can be supplied, preceding the length and color values. If present, this keyword causes the shadow to be drawn inside the element.

```
-webkit-box-shadow: 0 0 20px #333 inset;
-moz-box-shadow: 0 0 20px #333 inset;
box-shadow: 0 0 20px #333 inset;
```

text-shadow has the same set of properties as well.

It allows you to immediately apply depth to your elements. We can apply multiple shadows by using a comma as a separator.

text-shadow

```
color: #fff /* text color to white */
text-shadow: 0 0 50px #333;
```

Text shadows are like box shadows except that they are shadows for text rather than the whole element. Luckily, there is no vendor prefix necessary for text shadow.

The options for text shadow are the same as for box-shadow except that there is *no inset* text shadow support. As with box shadows, it is possible to have multiple text shadows just by separating them with commas. Here is an example that creates a flaming text effect.

```
text-shadow: 0 0 4px #ccc,
             0 -5px 4px #ff3,
             2px -10px 6px #fd3,
```

```
  -2px -15px 11px #f80,
             2px -18px 18px #f20;
```

Gradients

Just as you can declare the background of an element to be a solid color in CSS, you can also declare that background to be a gradient. Using gradients declared in CSS, rather than using an actual image file, is better for control and performance.

Gradients are typically one color that fades into another, but in CSS you can control every aspect of how that happens, from the direction to the colors (as many as you want) to

```
where those color changes happen.
.gradient {
  /* can be treated like a fallback */
  background-color: red;

  /* will be "on top", if browser supports it */
  background-image: linear-gradient(red, orange);

  /* these will reset other properties, like background-position, but it
does know what you mean */
  background: red;
  background: linear-gradient(red, orange);
}
```

Linear Gradient

Perhaps the most common and useful type of gradient is the linear-gradient(). The gradient "axis" can go from left-to-right, top-to-bottom, or at any angle you choose. Not declaring an angle will assume top-to-bottom. To make it left-to-right, you pass an additional parameter at the beginning of the linear-gradient() function starting with the word "to", indicating the direction, like "to right". This "to" syntax works for corners as well. You aren't limited to just two colors either. In fact you can have as many comma-separated colors as you want. You can also declare where you want any particular color to "start". Those are called "color-stops". Say you wanted yellow to take up the majority of the space, but red only a little bit in the beginning, you could make the yellow color-stop pretty early:

```css
.gradient {
  height: 100px;
  background-image:
    linear-gradient(
      to right,
      red,
      yellow 10%
    );
}
```

Gotchas

- We tend to think of gradients as fading colors, but if you have two color stops that are the same,This can be useful for declaring a full-height background that simulates columns.

- There are three different syntaxes that browsers have supported:

    - Old: original WebKit-only way, with stuff like from() and color-stop()

    - Tweener: old angle system, e.g. "left"

    - New: new angle system, e.g. "to right"

The way degrees work in the OLD vs NEW syntax is a bit different. The OLD (and TWEENER - usually prefixed) way defines 0deg and left-to-right and proceeds counter-clockwise, while the NEW (usually unprefixed) way defines 0deg as bottom-to-top and proceeds clockwise.

```
OLD (or TWEENER) = (450 - new) % 360
or even simpler:
NEW = 90 - OLD OLD = 90 - NEW
like:
OLD linear-gradient(135deg, red, blue) NEW linear-gradient(315deg, red,
blue)
```

Radial Gradients

Radial gradients differ from linear in that they start at a single point and emanate outwards. Gradients are often used to simulate a lighting, which as we know isn't always straight, so they can be useful to make a gradient seem even more natural.

The default is for the first color to start in the (center center) of the element and fade to the end color toward the edge of the element. The fade happens at an equal rate no matter which direction. The default gradient shape is an ellipse.

Gotchas

```
There are again three different syntaxes that browsers have supported:
Old: Prefixed with -webkit-, stuff like from() and color-stop()
Tweener: First param was the location of the center. That will completely
break now in browsers that support new syntax unprefixed, so make sure any
tweener syntax is prefixed.
New: Verbose first param, like "circle closest-corner at top right"
It is recommended to used autoprefixer like postcss to handle vendor
prefixes to make it work across different browsers consistently.
```

-

Repeating Gradients

The size of the gradient is determined by the final color stop. If that's at 20px, the size of the gradient (which then repeats) is a 20px by 20px square.

```
.repeat {
  background-image:
    repeating-linear-gradient(
      45deg,
      yellow,
      yellow 10px,
      red 10px,
      red 20px /* determines size */
    );
```

}

They can be used with both linear and radial varieties.There is a trick, with non-repeating gradients, to create the gradient in such a way that if it was a little tiny rectangle, it would line up with other little tiny rectangle versions of itself to create a repeating pattern.

References:

1. https://www.w3schools.com/css/css_syntax.asp
2. https://www.w3schools.com/css/css_selectors.asp
3. https://www.w3schools.com/cssref/sel_class.asp
4. https://www.w3schools.com/cssref/sel_id.asp
5. https://www.w3schools.com/cssref/sel_all.asp
6. https://www.w3schools.com/cssref/sel_element.asp
7. https://www.w3schools.com/css/css_pseudo_classes.asp