

- ▼ Adding Style
 - ▼ Terminologies
 - Selector
 - Declaration Block
 - Where to Include CSS in HTML
 - Common Declaration
- ▼ Selectors
 - Simple Selector
 - pseudo element
 - Combinators
 - Selector list (,)
- ▼ Cascading
 - 1. Compare Origin and Importance
 - 2. Compare Specificity
 - 3. Compare Order of Appearance
 - Application
 - Inheritance
- ▼ CSS Property Computation Process
 - 1. confirm declaration value
 - 2. Cascading Conflict
 - 3. Use Inheritance
 - 4. Use default values
- CSS Box Model

Adding Style

CSS Syntax

Terminologies

```
<body>
  <h1 class="red">
    Adding Style
  </h1>
  <p id="test">This is a paragraph</p>
  <p class="red big-center">This is a paragraph</p>
  <p class="big-center">This is a paragraph</p>
</body>
```

```
/*Element Selector*/
h1 {
  color: red;
  background-color: lightblue;
  text-align: center;
}
/*ID Selector*/
#test {
  color: blue;
}

/*Class Selector*/
.red {
  color: red;
}
```

CSS rulesets = selector (h1) + declaration block ({...})

Selector

Selector: Select elements

1. ID selector (#xxx): select element with corresponding iD. - only select one as only 1 id per element.
2. Element selector (element): select all the similar elements. - too broad
3. Class selector (.xxx) - select element with corresponding class name. - flexible
4. etc.

Declaration Block

written inside curly braces {}

Each declaration block contains multiple declarations or properties representing certain styles.

Where to Include CSS in HTML

How CSS is structured

Note: We can put CSS in body element. But performance wise, prefer putting css in head element, loading style first before rendering webpage.

1. External Stylesheet (recommended)
write inside separate .css file and import via `link`
2. Internal Stylesheet
write inside `style` element
< 200 lines of code
3. Inline Styles
write inside element
might be useful when working with JavaScript.

Benefits using external stylesheet:

1. resolves style redundancy
2. good for browser caching, improve loading/ response time
3. follow separation of concerns (HTML, CSS, JS) - easy to read and maintain

Benefits using internal stylesheet:

1. less documents needed.
2. increase the first loading time/ rendering speed.

Note: One webpage can have more than one `<link>` element.

```
<!--External Stylesheet-->
<head>
  <link rel="stylesheet" href="CSS Path">
</head>

<!--Internal Stylesheet-->
<head>
  <style>
    /* CSS*/
    p {
      color: red;
    }
  </style>
</head>

<!--Inline Styles-->

<body>
  <p style="color: red">This is paragraph</p>
</body>
```

Common Declaration

1. color

set color of element content - text

named-color value: pre-defined color name

```
/*Green*/
color: red;
```

Three Primary Color, Color Value: three primary color (red, green, blue), each color value ranged from 0 - 255

```
/*rgb*/  
color: rgb(0, 255, 0);  
  
/*HEX  
* base 16 => 16**2 = 255  
*/  
color: #008c8c;
```

orange red: #ff4400, #f40

black: #000000, #000

white: #ffffff, #fff

red: #ff0000, #f00

green: #00ff00, #0f0

blue: #0000ff, #00f

purple: #f0f

aqua: #0ff

yellow: #ff0

gray: #ccc

2. background-color

set background color of an element.

3. font-size

a. pixel, px: absolute unit, understand as how many pixel that the height of character occupied.

b. em: relative length unit, based on parent element font-size.

Every elemnt must have font-size, if not declared, it will use font-size of its parent element (e.g., html element), otherwise, use base font-size inside browsers.

Default browser font-size: 16px


Note: relative unit is converted to absolute unit after computation.

Settings

Appearance

Theme

Sea Foam



Reset to default

Mode

Light

Show home button

https://www.youtube.com/

☒

☐ New Tab page

☒ https://www.youtube.com/

Show bookmarks bar

☒

Show images on tab hover preview cards

☒

Side panel

☒ Show on right

☐ Show on left

Font size

Medium (Recommended)

Customize fonts

Page zoom

100%

4. font-weight

specify font weight, set by named value or number value.

5. font-family

effective only if user have the font-family in their pc.

specify more than one font-family.

sans-serif:

- font without serif, small decorative pieces on the end of each character.
- good for web

```
font-family: Arial, sans-serif; /*sans-serif as fallback if the preceding font family is not present*/
```

6. font-style

sets whether a font should be styled with a normal, italic, or oblique face from its font-family.

By default, `<i>` , `` , their font-style is *italic*, in real life, commonly use for icon.

By default, `` font-style is **bold**

7. text-decoration

sets the appearance of decorative lines on text

`a` element - text-decoration: underline

`del` element: represents text that has been deleted during document editing - text-decoration: line-through

`s` element: represent texts that are no longer relevant or no longer accurate - text-decoration: line-through.

9. text-indent

set the length of empty space (indentation) that is put before lines of text in an element.

10. line-height

sets the height of a line box

the greater the value, the greater the distance between each line

use to set line-height as the container height, and align text center vertically

```
/*to vertically align single-line text*/
height: 50px;
line-height: 50px;
```

line-height can be set by pure numeric value, which is relative to current element font-size.

```
line-height: 50px;
```

```
line-height: 1.5; /*recommended*/
```

11. width

12. height

13. letter-spacing

sets the horizontal spacing behavior between text characters

14. text-align

sets the horizontal alignment of the inline-level content inside a block element or table-cell box.

Selectors

Selector: Precisely select any element we want

Simple Selector

a selector with a single component

1. ID selector

matches an element based on the value of the element's id attribute

2. class selector

matches elements based on the contents of their class attribute

3. universal selector

`*` , matches elements of any type.

```
* {}
```

4. element selector

```
p {}
```


5. attribute selector

matches elements based on the element having a given attribute explicitly set, with options for defining an attribute value or substring value match.

```
[attr]
```

```
[href] {}
```

6. pseudo class

selects elements that are in a specific state

:link : represents an element that has not yet been visited.

:visited : represents an element has been visited.

:hover : when hover over an element with mouse pointer

:active : when clicking an element with mouse pointer

```
:hover {}  
/*When hovering over an element*/  
a:hover {}  
/*when click an element*/  
a:active {}
```

Note: the order matters.

pseudo element

a keyword added to a selector that lets you style a specific part of the selected element(s)

```
::pseudoelement
```

Examples:

```
::before
```

```
::after
```

Note:

The pseudo-element must appear after all the other components in the complex or compound selector in which it appears.

```
p:hover::first-line /*valid*/
```

```
p::first-line > * /*invalid*/
```

```
p::first-line:hover /*invalid*/
```

Combinators

1. AND

```
a:hover {  
  color: red;  
}
```

```
p.green{  
  color: green;  
}
```

2. descendant combinator (" ") **

works for child element or any descendant, not necessary the direct child.

```
.red li {  
  color: yellowgreen;  
}
```

```
.abc * {  
  color: blue;  
}
```

3. child combinator (>)

matches only those elements matched by the second selector that are the direct children of elements matched by the first.

```
.a > .bcd {  
  color: green;  
}
```

4. next-sibling combinator (+)

matches only those elements matched by the second selector that are the next sibling element of the first selector.

```
.special + li {  
  color: #008c8c;  
}
```

5. subsequent-sibling combinator

matches all the subsequent sibling (~)elements

```
.special ~ li {  
  color: chocolate;  
}
```

Note: There's no previous sibling selector due to internal browser rendering process.

Selector list (,)

Syntactic sugar: make the code easier to read or write.

separates multiple selectors within the same declaration.

prevent redundancy



```
.special~li, p {  
  color: #008c8c;  
}
```



Cascading


CSS / declaration conflict: when the same style is applied to the same element multiple time.

cascade: a process to resolve CSS conflict, browser deals with it automatically. (specificity calculation)

```

.act {                                     cascading.html:14
  color:  white;
  background-color:  red;
}

a {                                       cascading.html:8
  color:  red;
  text-decoration:  none;
  font-style: italic;
}

a:-webkit-any-link {                   user agent stylesheet
  color: webkit-link;
  cursor: pointer;
  text-decoration:  underline;
}

```

1. Compare Origin and Importance

Importance from top to bottom:

1. !important in author style sheet (developer's style sheet)

Note: don't use.

```

a{
  color: red !important; /*!important takes precedence*/
  text-decoration: none;
  font-style: italic;
}

```

```

.act {
  color: white;
  background-color: red;
}

```

2. styles in the author style sheet.
3. user agent stylesheet (browser's style sheet)

2. Compare Specificity

based on CSS selector.

General rule: the declaration with the highest specificity (narrower scope) wins.

```
a{
  color: red;
  text-decoration: none;
  font-style: italic;
}
```

```
.act {
  color: white;
  background-color: red;
}
```

class selector (.act) > element (a)

Specific rule: through selector, calculate a four-digit number (x x x x)

Note: Don't consider rounding for each x.

People tested rounding occurs when 255.

1. thousands place: if inline style, mark as 1, else 0.

```
<a style="color: blue;"> abc </a>
```

2. hundreds: the number of id selectors

```
#mylink {
  /*0100*/
  color: gray;
}
```

```
#body #mylink {
  /*0200*/
  color: gray;
}
```

3. tens: the number of class selector, attribute selector, pseudo class.

```
.act {  
  /*0010*/  
  color: white;  
  background-color: red;  
}
```

4. ones: the number of element selector, pseudo element selector

```
a {  
  /*0001*/  
  color: #fff;  
}
```

```
body a {  
  /*0002*/  
  color: #fff;  
  background-color: red;  
}
```

Note: we don't change other people style, we cover them with our style that has higher specificity.

3. Compare Order of Appearance

Rule: the last declaration in the style order is applied.

works with similar specificity.

Application

1. CSS Reset Style Sheet - considered as author stylesheet

an author stylesheet that 'reset' all of the default browser styles.

has more importance than user agent stylesheet.

Why is reset important?

Different browser has their own user stylesheet, which may behave differently.

common reset stylesheet: normalize.css, reset.css, meyer.css

[meyer css reset](#)

2. :link > :visited > :hover > :active (love hate)

Inheritance

Inheritance happens after cascading.

child elements inherit CSS properties from their parent element.

Generally, font-related properties are inheritable.

Check MDN.

e.g.,

font-size, font-style, font-family, text-align etc. can be inherited

background color is not inheritable.

```
body {  
  font-family: Arial, Helvetica, sans-serif;  
  background-color: green;  
}
```

Note: p element background color is transparent, that's why we can see the body background color behind p elements.

Question: How do inheritance and cascading work together and resolve conflict?

Inheritance comes after cascading, thus there's no conflict. Inheritance only works with properties without values, those having values in the previous cascading step is not concerned in the inheritance step.

Question: Isn't author stylesheet winning over user agent stylesheet?

`-webkit-match-parent` : match parent value.

Inheritance comes after cascading and only concern those properties without values.

Before cascading step, user agent already set the value for text-align property of li element

`-webkit-match-parent` .

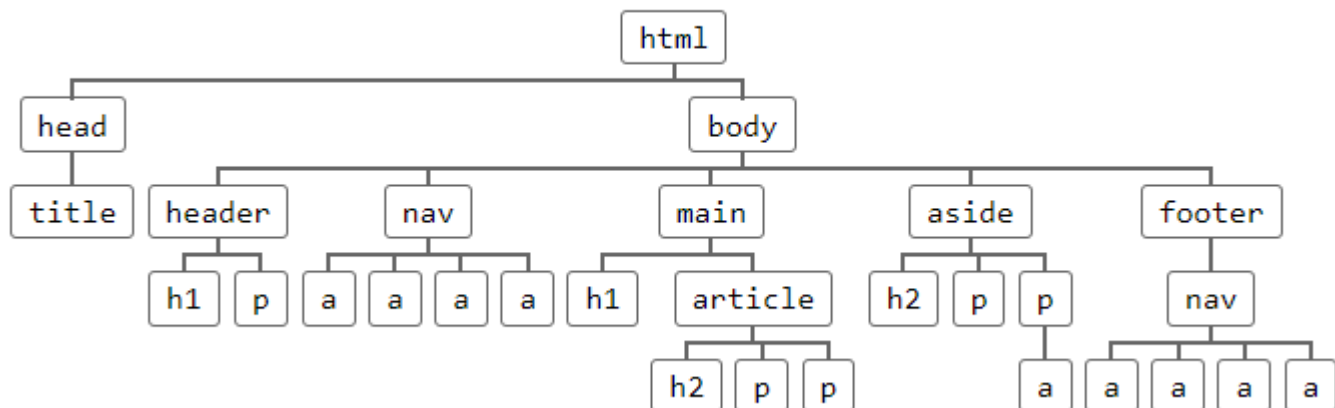
Thus, in the inheritance step, although it is possible to inherit value (text-align: center) from parent element, it is not something to be concerned here as `li` already has value in text-align that comes from the previous steps.

```
li {                                user agent stylesheet
  display: list-item;
  text-align: -webkit-match-parent;
}

body {                               inheritance.html:8
  background-color: lavender;
  text-align: center;
}
```

CSS Property Computation Process

Example DOM structure diagram



Rendering process begins from top down, one element at a time:

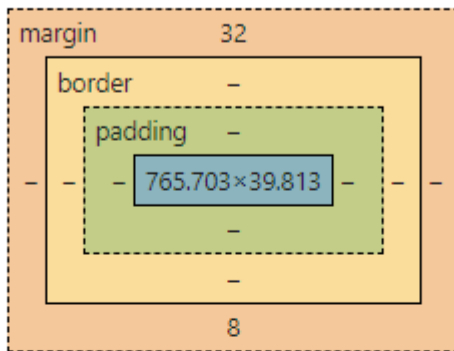
html --> head --> body

Prerequisite of rendering each element: each element's properties must have values.

computation --> rendering --> computation --> rendering --> ...until everything is rendered.

The property value computation process involves determining value of all its properties, starting from having no values to eventually having value.

- no values --> values



Filter	<input checked="" type="checkbox"/> Show all <input type="checkbox"/> Group
► accent-color	■ rgb(0, 133, 242)
align-content	normal
align-items	normal
align-self	auto
alignment-baseline	auto
all	
animation-composition	replace
animation-delay	0s
animation-direction	normal
animation-duration	0s
animation-fill-mode	none
animation-iteration-count	1
animation-name	none

1. confirm declaration value

declarations of stylesheets without conflict, their values will be taken as CSS property value.

```
/*CSS properties*/
background-color:;
color: red;
text-align: ;
font-size: ;
font-weight: bold;
display: block;
etc...
```

```
<h1 class="red">Lorem</h1>
```

```
/* author stylesheet*/
```

```
.red {  
  color : red; /*take*/  
  font-size: 40px;  
}
```

```
h1 {  
  font-size: 26px;  
}
```

```
div h1.red {  
  font-size: 3em;  
  font-size: 30px;  
}
```

```
/* user agent stylesheet*/
```

```
h1 {  
  display: block; /*take*/  
  font-size: 2em;  
  font-weight: bold; /*take*/  
}
```

2. Cascading Conflict

declaration with conflict, resolve using *cascading rules*:

1. compare origin and importance.
2. compare specificity.
3. compare the order of appearance.

```

/*CSS properties*/
background-color;;
color: red;
text-align: ;
font-size: 30px;
font-weight: bold;
display: block;
etc...

/* author stylesheet*/
.red {
  /*0010*/
  color : red; /*take*/
  font-size: 40px; /*drop*/
}

h1 {
  /*0001*/
  font-size: 26px; /*drop*/
}

div h1.red {
  /*0012*/
  font-size: 3em; /*drop*/
  font-size: 30px; /*take - who comes latter wins*/
}

/* user agent stylesheet*/
h1 {
  /*0001*/
  display: block; /*take*/
  font-size: 2em; /* drop*/
  font-weight: bold; /*take*/
}

```

3. Use Inheritance

If properties of an element *still has no values*, if inheritance is possible, it will inherit values from its parent element.

Note:

1. Only consider properties without values.
2. parent element has done rendering and has all the property values, thus the child element can inherit from them.

`inherit` : force inheritance, take value from parent element and apply to element using this keyword. It can be applied to any CSS property.

`initial` : default, set the property with default value.

Inheritance only from direct parents. If it is from other branch parent, it is not counted.

```
/*CSS properties*/  
background-color;;  
color: red;  
text-align: center; // from inheritance  
font-size: 30px;  
font-weight: bold;  
display: block;  
etc...
```

```

/*Parent CSS declarations*/
div {
    color: green;
    background-color: gray;
    text-align: center; /*will be inherited*/
    font-size: 20px;
    font-weight: normal;
    display: block;
    etc...
}

/* author stylesheet*/
.red {
    /*0010*/
    color : red; /*take*/
    font-size: 40px; /*drop*/
}

h1 {
    /*0001*/
    font-size: 26px; /*drop*/
}

div h1.red {
    /*0012*/
    font-size: 3em; /*drop*/
    font-size: 30px; /*take - who comes latter wins*/
}

/* user agent stylesheet*/
h1 {
    /*0001*/
    display: block; /*take*/
    font-size: 2em; /* drop*/
    font-weight: bold; /*take*/
}

```

4. Use default values

If some properties still have no values, use default (initial value).

check MDN for initial value of each CSS property.

```
/*CSS properties*/  
background-color: transparent; // *** initial value ***  
color: red;  
text-align: center ;  
font-size: 30px;  
font-weight: bold;  
display: block;  
etc...
```

CSS Box Model