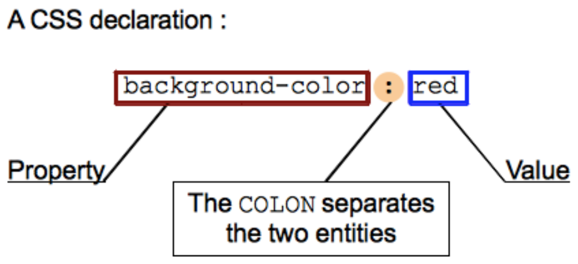
CSS – Cascading Style Sheets

Cascading Style Sheets (CSS) is a **stylesheet language** used to **describe the presentation of a document** written in HTML or XML. CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

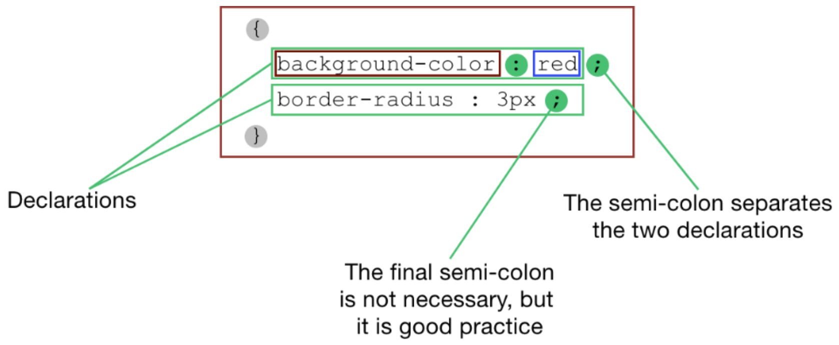
CSS is one of the core languages of the open Web and is standardized across Web browsers according to the W3C specification. From CSS3 instead of versioning the CSS specification, W3C now periodically takes snapshot of the latest stable state of the CSS specification.

Syntax:

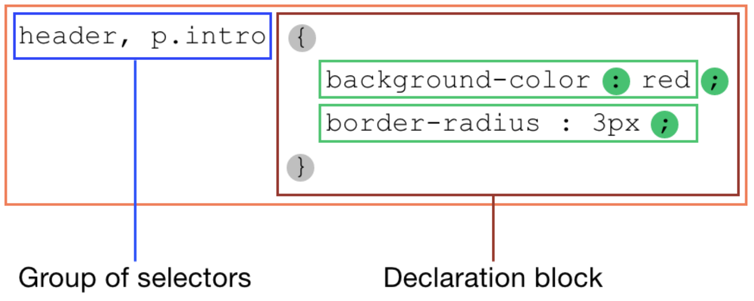
- **CSS Declarations:** A **property and value pair** is called a declaration, and any **CSS engine** calculates which declarations apply to every single element of a page in order to appropriately lay it out, and to style it.



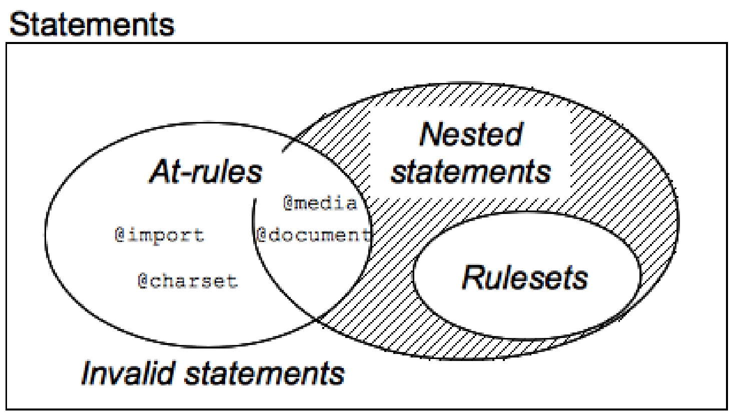
- **CSS Declaration Blocks:** Declarations are **grouped** in blocks, that is in a structure delimited by an opening brace, **{**, and a closing one, **}**


{
background-color : red ;
border-radius : 3px ;
}
Declarations
The semi-colon separates the two declarations
The final semi-colon is not necessary, but it is good practice

- **CSS Rulesets:** Each (valid) **declaration block** is preceded by one or more comma-separated **selectors**, which are conditions selecting some elements of the page. **A selector group and an associated declarations block**, together, are called a **ruleset**, or often simply a **rule**.



- **CSS Statements:**
  Two types – Rulesets & At-rules



Integration with HTML:

3 ways:

- External Style Sheet
- Internal Style Sheet
- Inline Style Sheet

Comments:

A CSS comment is used to **add explanatory notes** to the code or to **prevent the browser from interpreting specific parts of the style sheet**. Comments can be used on a single line, or traverse multiple lines.

Example:

```
/* A one-line comment */

/*
A comment
which stretches
over several
lines
*/
```

Selectors:

Selectors **allow styles to be conditional** based on various features of elements within the DOM.

- **Basic Selectors:**

a. Universal Selector:

*	Matches elements of **any type**.

b. Type Selector:

element	Matches elements by node name, it selects **all elements of the given type** within a document.

c. Class Selectors:

.class_name	Matches elements based on the **contents of their class attribute**.

d. ID Selectors:

#id_value	Matches an element based on the **value of the element's id attribute**.

e. Attribute Selectors:

[attr]	Matches elements with an attribute name of **attr**
[attr=value]	Matches elements with an attribute name of **attr** whose value is **exactly value**.
[attr~=value]	Matches elements with an attribute name of **attr** whose value is a whitespace-separated list of words, one of which is exactly **value**.
[attr^=value]	Matches elements with an attribute name of **attr** whose value is prefixed(preceded) by **value**.
[attr$=value]	Matches elements with an attribute name of **attr** whose value is suffixed(followed) by **value**.

- **Grouping Selectors:**
  a. **Selector list:**

element1, element2, element3	Selects **all the matching nodes**. A single unsupported selector in a selector list invalidates the whole rule.

- **Combinators:**
  a. **Child Combinator:**

selector1 > selector2	Matches only **those elements matched by the second selector** that are the direct children of elements matched by the first.

  b. **Descendant Combinator:**

selector1 selector2	Combines two selectors such that **elements matched by the second selector are selected** if they have an ancestor element matching the first selector.

- **Pseudo-Selectors:**
  a. **Pseudo Class:**

It is a keyword added to a selector that specifies **a special state** of the selected element(s).

Syntax:

```
selector:pseudo-class{
      property: value;
      … … …
}
```

:link	Represents an element that has **not yet been visited**.
:visited	Represents links that the user has **already visited**.
:hover	Triggers when the user **hovers over an element** with the cursor.
:active	Represents an element that is being **activated by the user** (activated means user presses down the primary mouse button).
:checked	Represents any radio, checkbox, option element that is checked or toggled to an **on state**.
:disabled	Represents any **disabled** element (meaning the element can't be clicked/typed/selected/focused).
:focus	Represents an element that has **received focus**. (meaning user has clicked on that element for use)
:read-only	Represents an element that is **not editable** by the user.
:required	Represents any <input>, <select>, or <textarea> element that has the **required attribute** set on it.
:invalid	Represents any <input> or other <form> element whose **contents fail to validate.** For example: invalid email field or, invalid input text that doesn't match with what is specified in the pattern attribute.

:valid	Represents any <input> or other <form> element whose **contents validate successfully**. For example, valid email field or, valid input text that matches with what is specified in the pattern attribute.
:not(selector_list)	Represents elements that **do not match a list of selectors**. It prevents specific items from being selected.
:nth-child(child_no) **child_no = 1,2,3, … …** **child_no= 3n+1 for n =0,1,2,3** **child_no= even, odd**	Matches elements based on their **position in a group of siblings** (counting starts from 1)
:nth-last-child(child_no) **child_no = 1,2,3, … …** **child_no= 3n+1 for n =0,1,2,3** **child_no= even, odd**	Matches elements based on their **backward position in a group** of siblings.

b. **Pseudo Elements:**

A CSS pseudo-element is a keyword added to a selector that **lets you style a specific part** of the selected element(s).

Syntax:

```
selector::pseudo-element{
      property: value;
      … … … …
}
```

Frequently used Pseudo-elements list:

::after, ::before, ::first-line, ::first-letter

Precedence:

- **Cascading:**

The **cascade** is an algorithm that defines how to combine property values originating from different sources. It lies at the core of CSS and thus named: **Cascading Style Sheets.**

Only **CSS declarations**, that is **property/value pairs**, participate in the cascade. Generally Style Sheet comes from 3 different sources:

- **User-agent stylesheets –** Browser has a basic style sheet that gives a default style to any document.
- **Author stylesheets –** These style sheets define styles as part of the design of a given webpage.
- **User stylesheets –** Website users can choose to override styles in many browsers. **Ex.** Stylebot

Cascade Order (low to High):

user agent	normal
user	normal
author	normal
animations	
author	!important
user	!important
user agent	!important
transitions	

- **Inheritance:**

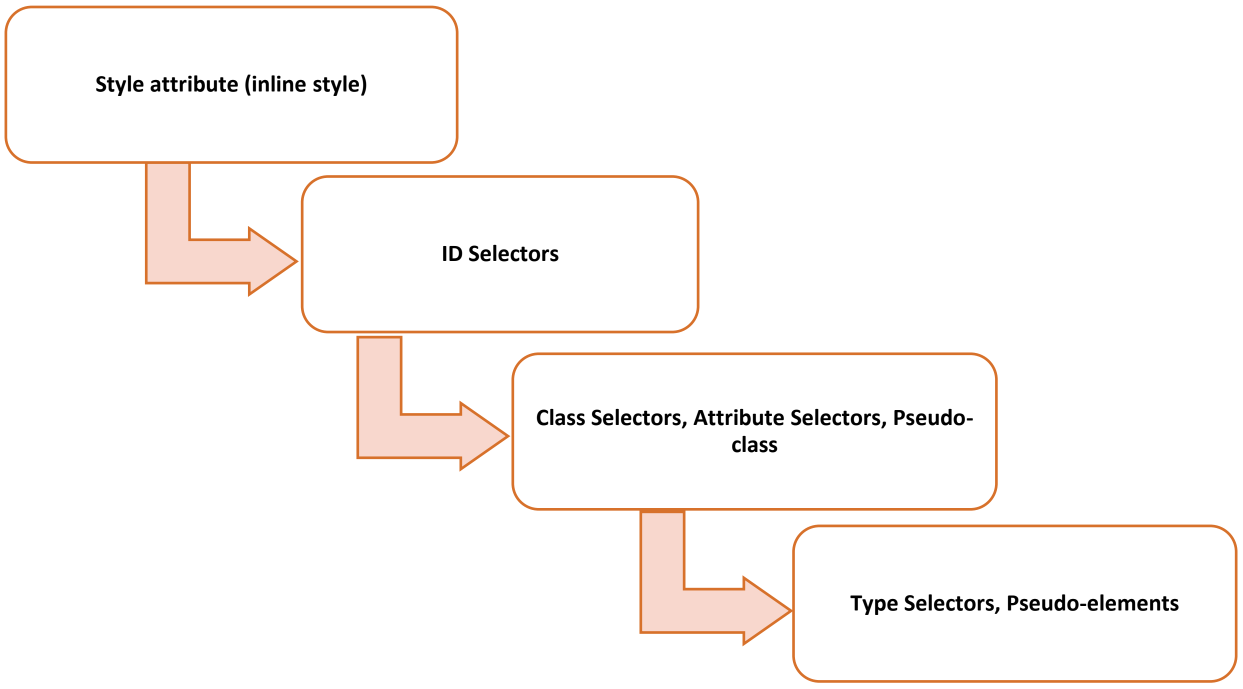
  It controls what happens when **no value is specified** for a property on an element.

Inherited properties	This type of properties (of the child) are set to the computed value of the parent element. **Example:** color, direction, font-*, list-*, text-*, word-* etc.
Non-inherited properties	This type of properties (of the child) are set to the initial value of the property. **Example:** width, height, margin, padding, border, background-*, float, position, overflow, z-index, display etc.

- **Specificity:**

  When same element is targeted by multiple declarations then we need to use specificity rules to detect which rules will be applied.

The specificity order (high to low):


Style attribute (inline style)
ID Selectors
Class Selectors, Attribute Selectors, Pseudo-class
Type Selectors, Pseudo-elements

Box Model:

When laying out a document, the browser's rendering engine represents **each element as a rectangular box** according to the standard CSS basic box model. CSS determines the size, position, and properties (color, background, border size, etc.) of these boxes.

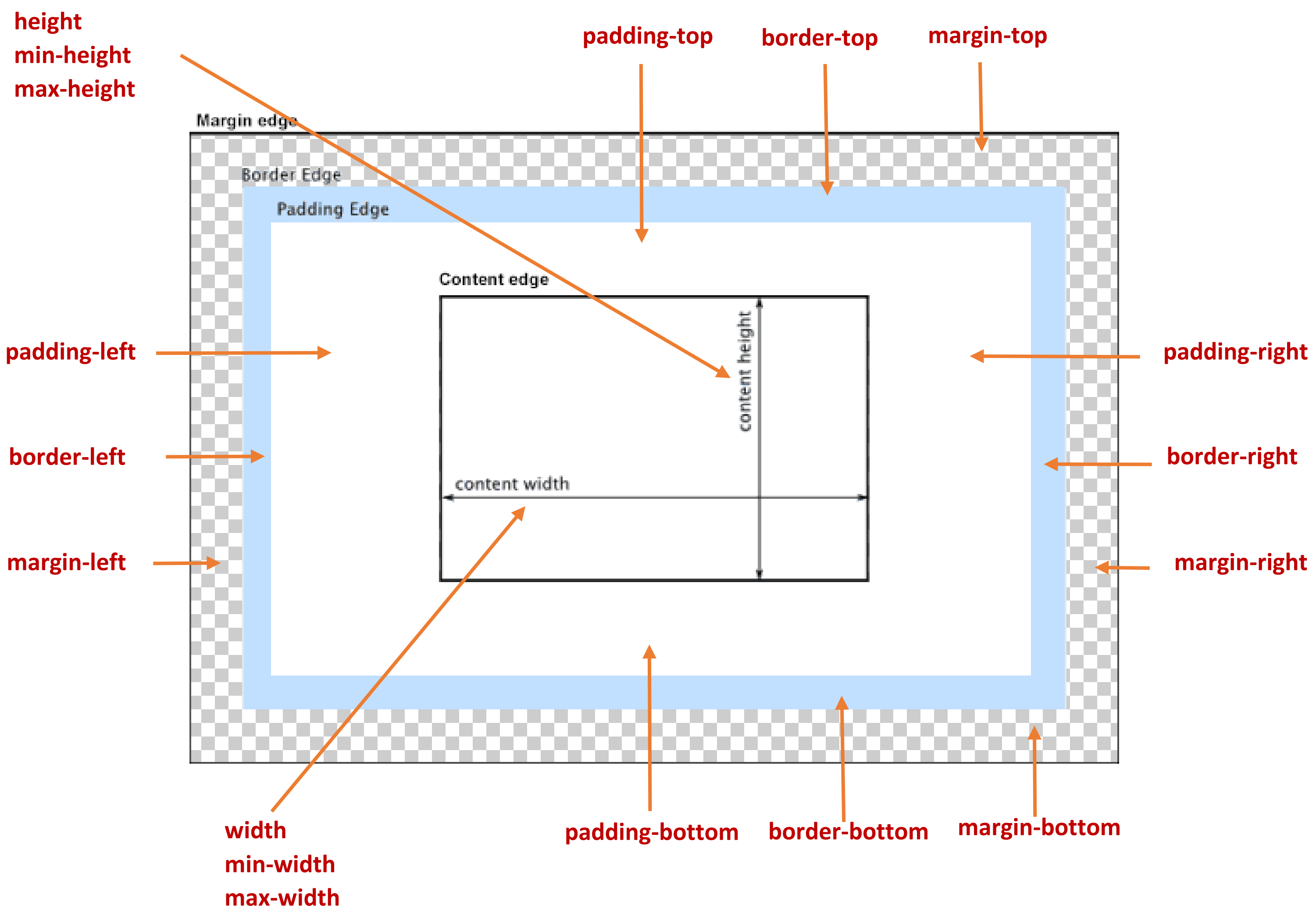
Every box is composed of four parts (or areas), defined by their respective edges:

- **Content area -** The content area, bounded by the content edge, contains the "real" content of the element, such as text, an image, or a video player. Its dimensions are the content width (or content-box width) and the content height (or content-box height). It often has a background color or background image.

- **Padding area -** The padding area, bounded by the padding edge, extends the content area to include the element's padding. Its dimensions are the padding-box width and the padding-box height.

- **Border area -** The border area, bounded by the border edge, extends the padding area to include the element's borders. Its dimensions are the border-box width and the border-box height.

- **Margin area -** The margin area, bounded by the margin edge, extends the border area to include an empty area used to separate the element from its neighbors. Its dimensions are the margin-box width and the margin-box height.



Property Names	Possible Values
width, min-width, max-width	**length(absolute value):** `10vw, 20vh, 50px`
height, min-height, max-height	
	percentage(value relative to the containing block
padding-top, padding-right, padding-bottom, padding-left	**size):** `20%, 50%, 80%`
border-width	**auto:** Browser will calculate and select a width for the specified element
margin-top, margin-right, margin-bottom, margin-left	
	calculated value: calc(100% - 30px), calc(10px*2),
outline-width	min(50vw, 200px), max(20vh, 100px)

Shorthand Properties:

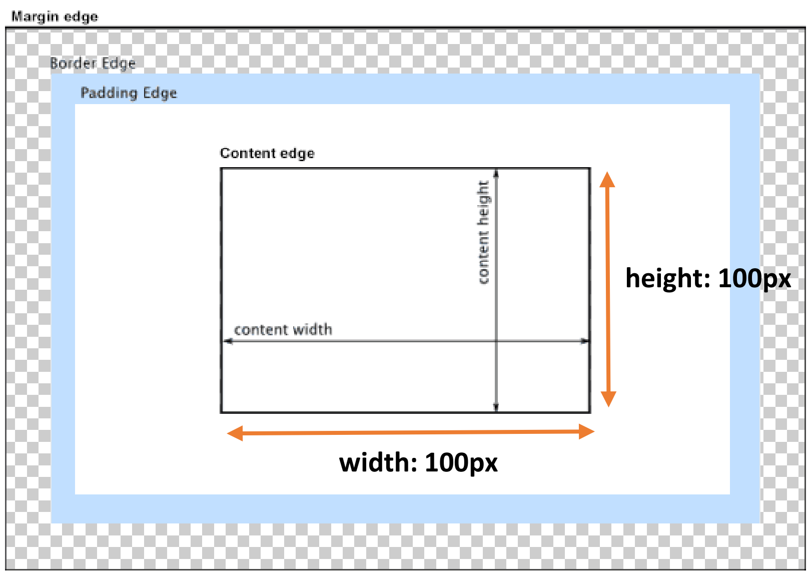
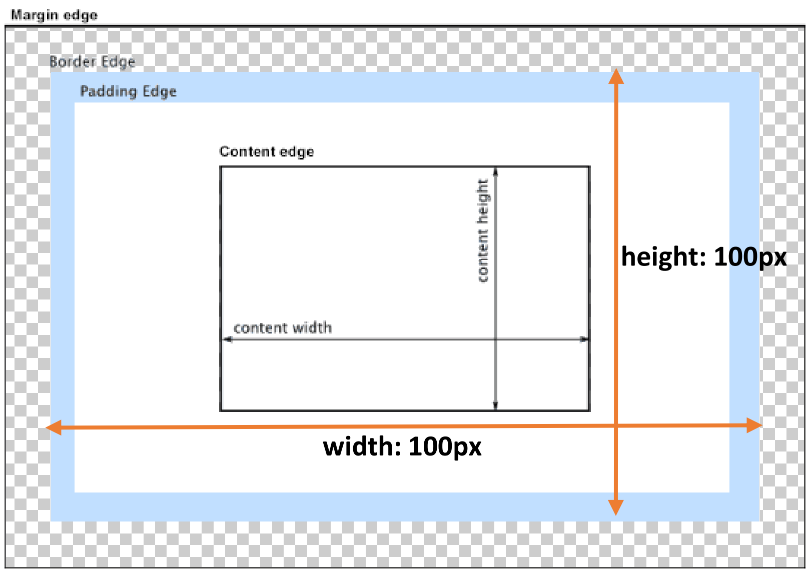
margin: top right bottom left	**margin:** 5px 10px 15px 8px
margin: top-bottom right-left	**margin:** 10px 5px
margin: top-right-bottom-left	**margin:** 5px
padding: top right bottom left	**padding:** 5px 10px 15px 8px
padding: top-bottom right-left	**padding:** 10px 5px
padding: top-right-bottom-left	**padding:** 5px
border-width: 1px/10%/2vw	
border-style: none/solid/dashed/dotted/inset	
border-color: blue	
in short,	
border: 1px solid blue	
border-radius: all-side	**border-radius:** 10px
border-radius: top-left_bottom-right top-right_bottom-left	**border-radius:** 10px 15px
border-radius: top-left top-right bottom-right bottom-left	**border-radius:** 1px 2px 4px 3px
outline: 2px solid gold	**Outline and borders are very similar**. The difference is outlines never take up space.

Sample Code Link:

Margin Collapsing:

The top and bottom margins of blocks are sometimes combined (collapsed) into a single margin whose size is the largest of the individual margins (or just one of them, if they are equal), a behavior known as margin collapsing. Note that the margins of floating and absolutely positioned elements never collapse.

Other Properties:

box-sizing: content-box / border-box	**content-box:**  Margin edge Border Edge Padding Edge Content edge content height content width height: 100px width: 100px **border-box:**  Margin edge Border Edge Padding Edge Content edge content height content width height: 100px width: 100px
box-shadow: [inset] offset-x offset-y [blur-radius] [spread-radius] color - adds shadow effects around an element's frame.	box-shadow: inset 2px 2px 2px 1px rgba(0, 0, 0, 0.2) box-shadow: 2px 2px 2px 1px rgba(0, 0, 0, 0.2) box-shadow: 0 10px 6px -6px #777
overflow: visible/hidden/scroll/auto **overflow-x:** visible/hidden/scroll/auto **overflow-y:** visible/hidden/scroll/auto - It sets what to do when an element's content is too big to fit in its block formatting context.	**visible:** content is not clipped **hidden:** content is clipped if necessary to fit the padding box **scroll:** always displays scrollbars irrespective of content overflow **auto:** if content overflows then shows the scroll bar

visibility: visible/hidden
display: none/inline/inline-block/block

visible: the element box is visible
hidden: the element box is invisible but still takes up the space

block: generates a block element box
inline: generates inline element box, the next element will be in the same line
inline-block: generates a block element box that will act like a single inline box
none: hides the element, also don’t take up space

---

background-color: red/#bbff00/rgb(117,190,218)/ transparent/rgba(255,255,128,.5)/#11ffee00

background-image: url(‘topimagepath’), url(‘backimagepath’)

background-repeat: repeat-x/repeat-y/ repeat/no-repeat

background-position: left top/left center/left bottom/right top/right center/right bottom/center top/center center/center bottom/10px 10px

background-size: contain/cover/auto/100px 100px/100% 100%

background-attachment: scroll/fixed

```
body{
        background-image: url(‘myimage.png’);
        background-repeat: no-repeat;
        background-attachment: fixed;
        background-size: cover;
        background-position: center center;
}
```

---

color: orange/#ffff32/rgb(255,255,50)
letter-spacing: 3px

text-align: center/left/right/justify
vertical-align: top/middle/bottom

text-decoration: none/underline/line-through
text-transform: uppercase/lowercase/capitalize

font-family: “Times New Roman”, Times, serif
font-size: 40px/120%/1.2em
font-style: normal/italic
font-weight: normal/bold

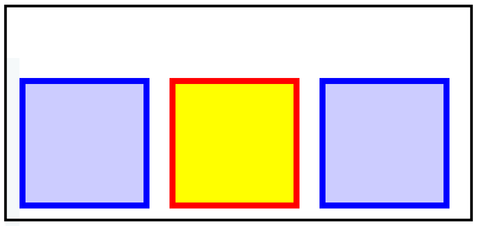
Positioning Elements:

The position CSS property sets how an element is positioned in a document. The **top, right, bottom, and left** properties determine the final location of the positioned elements.

position: static

- Positioned according to the normal flow of the document. The *top, right, bottom, left, z-index* properties have no effect.

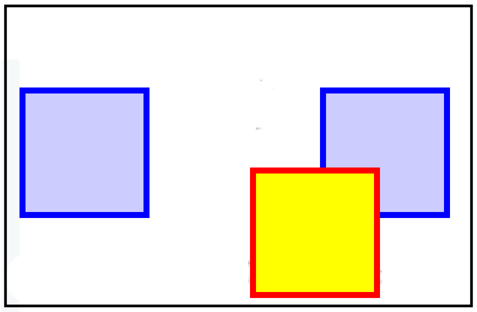
position: **static;**



position: relative

- Positioned according to the normal flow of the document and then offset relative to itself based on the values of top, right, bottom, left.

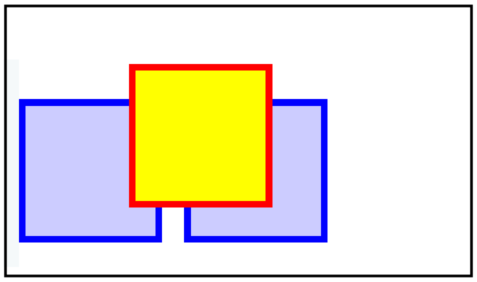
position: relative; **top:**40px; **left:**40px;



position: absolute

- The element is removed from the normal document flow, and no space is created for the element in the page layout.
- It is positioned relative to it's nearest ancestor element that has a position value other than static.
- It's final position is determined by the value of top, right, bottom, left.

position: absolute; **top:** 40px; **left:** 40px;



position: fixed

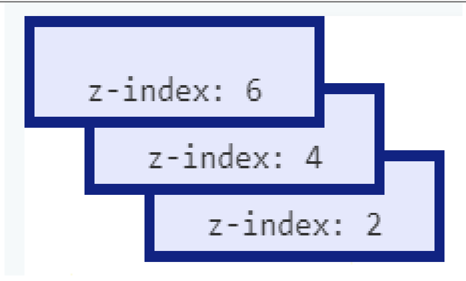
- The element is **removed from the normal document flow** and no space is created for the element in the page layout.
- It is positioned **relative to the initial containing block** established by the viewport.
- It's final position is determined by the values of top, right, bottom, left.

position: sticky

- The element is **positioned according to the normal flow** of the document and then offset relative to its nearest scrolling ancestor and containing block.

z-index: 2/4/6

- Sets the z-order of a positioned element and its descendants.
- higher z-index cover those with a smaller one



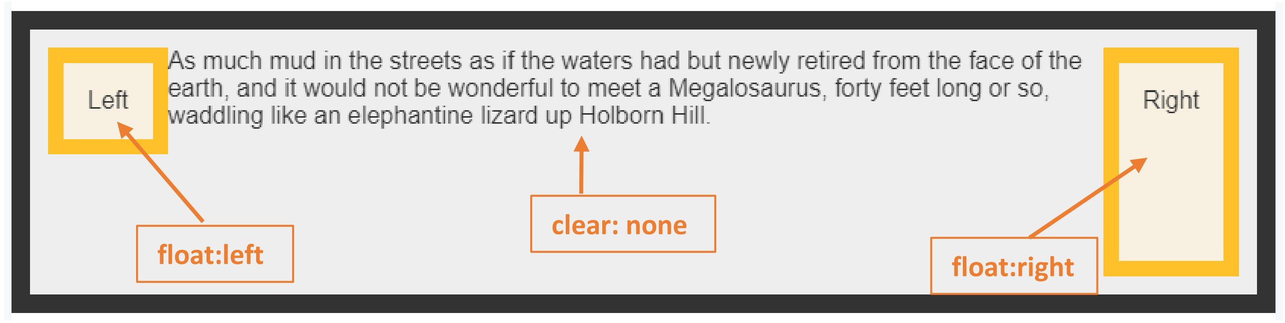
Float Property:

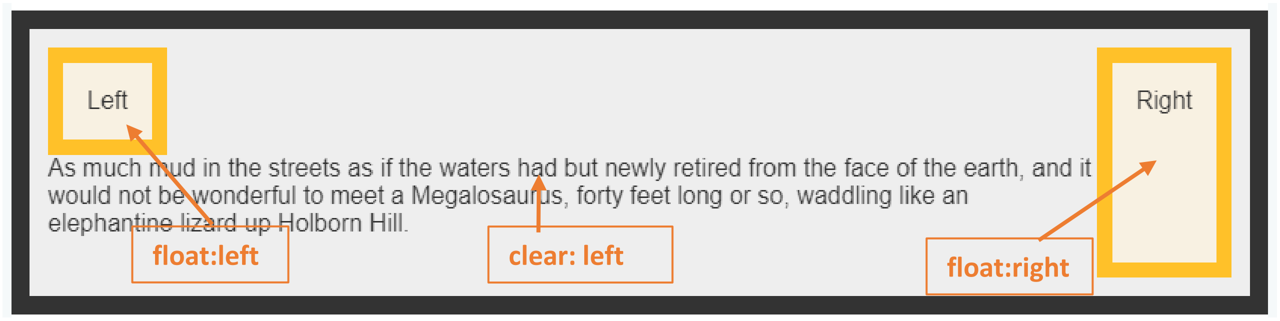
float: left/right/none

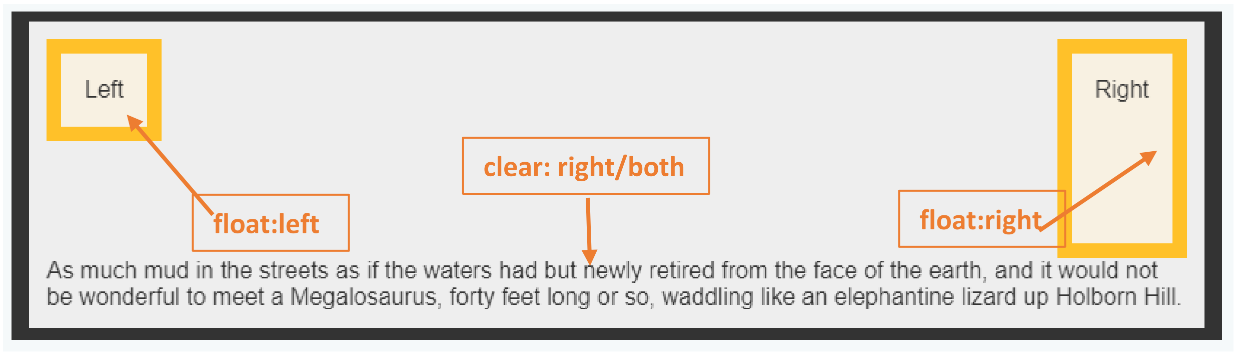
- It places an element on the left/right side its container, allowing text and inline elements to wrap around it.
- The element is removed from the normal flow of the page.

clear: left/right/both/none

- It sets whether an element must be moved below floating elements that precede it.


Left
As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.
Right
float:left
clear: none
float:right


Left
Right
As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.
float:left
clear: left
float:right


Left
Right
clear: right/both
float:left
float:right
As much mud in the streets as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn Hill.

References:

1. https://developer.mozilla.org/en-US/docs/Web/CSS/Reference
2. https://www.w3schools.com/css/
3. https://css-tricks.com
4. https://caniuse.com