CSS 3

What is CSS?

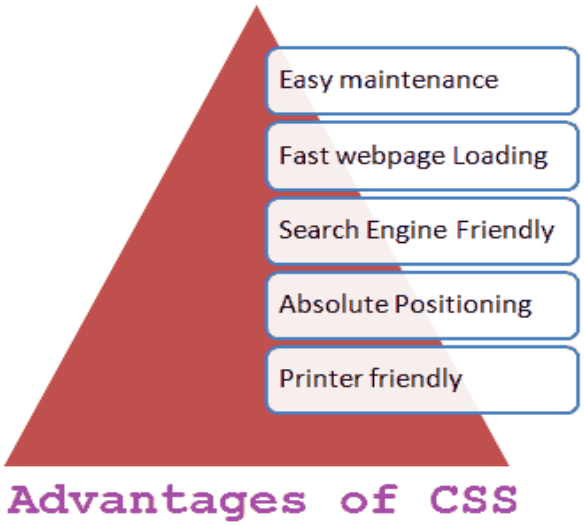
Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

- CSS handles **the look and feel part** of a web page.
- Using CSS, you can control the **color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used,** as well as a variety of other effects.
- CSS is easy to learn and understand but it provides a powerful control over the **presentation** of an HTML document.
- Most commonly, CSS is combined with the markup languages HTML or XHTML

Advantages of CSS

- **CSS saves time** - You can **write CSS once and then reuse the same sheet in multiple** HTML pages. You can define a style for each HTML element and apply it to as many web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all the elements in all the web pages will be updated automatically.

- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cellphones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.



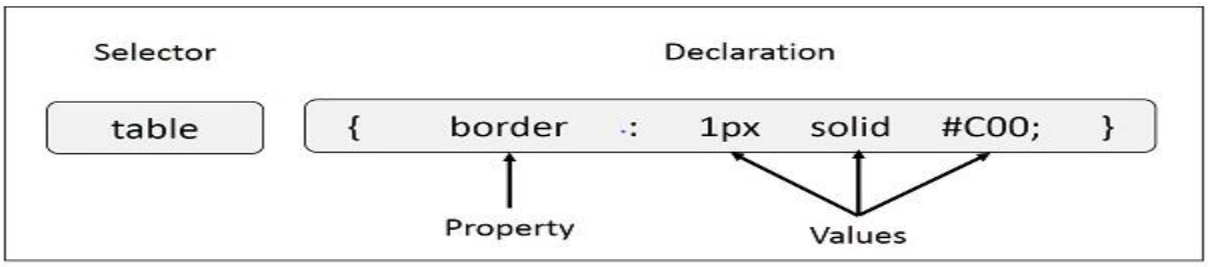
CSS SYNTAX:

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

- **Selector:** A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- **Property:** A property is a **type of attribute of HTML tag**. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border, etc.

- **Value**: Values are assigned to properties. For example, color property can have the value either red or #F1F1F1 etc.

CSS Style Rule Syntax as follows: **selector { property: value }**



TYPES OF CSS

There are 3 types of CSS. They are:

1. **External style sheet**
2. **Internal style sheet**
3. **Inline style sheet**

1.External (in a separate file)

- External style sheets are **separate files full of CSS instructions** (with the file extension .css).
- An external CSS stylesheet can be applied to **any number of HTML documents** by placing a **<link> element in each HTML document**.
- The **attribute rel of the <link> tag** has to be set to **"stylesheet",** and the **href attribute to the relative or absolute path to the stylesheet.**
- While using **relative URL paths is generally considered good practice**, absolute paths can be used, too. In HTML5 the type attribute can be omitted.
- It is recommended that the **<link> tag be placed in the HTML file's <head> tag so that the styles are loaded before the elements** they style. Otherwise, users will see a flash of unstyled content.

Example :

1. hello-world.html

```
<!DOCTYPE html>
 <html>
<head>
<meta charset="utf-8" />
 <link rel="stylesheet"  type="text/css"  href="style.css">
</head>
   <body>
<h1>Hello world!</h1>
<p>I ♥ CSS</p>
</body>
 </html>
```

2.style.css

```
h1 {
 color: green;
text-decoration: underline;
}

p {
 font-size: 25px;
 font-family: 'Trebuchet MS', sans-serif;
}
```

2.Internal (at the top of a web page document)

- Internal styles are **placed at the top of each web page document**, before any of the content is listed.
- This is the next best thing to external, because they're easy to find, yet allow you to **'override' an external style sheet** -- for that special page that wants to be a nonconformist!
- CSS enclosed in **<style></style> tags within an HTML document functions like an external stylesheet**, except that it lives in the HTML document it styles instead of in a separate file,
- and therefore can only be applied to the document in which it lives. Note that this element must be **inside the <head> element for HTML validation** (though it will work in all current browsers if placed in body).

Example:

```
<head>
    <style>
        h1 {
            color: green;
            text-decoration: underline;
           }
        p {
            font-size: 25px;
            font-family: 'Trebuchet MS', sans-serif;
           }
    </style>
</head>
<body>
```

```
    <h1>Hello world!</h1>
    <p>I ♥ CSS</p>
</body>
```

3.Inline (right next to the text it decorates)

- Inline styles are **placed right where you need them**, next to the text or graphic you wish to decorate.
- You can insert **inline styles anywhere in the middle of your HTML code**, giving you real freedom to specify each web page element.
- Use inline styles to apply styling to a specific element. Placing style rules in **a <style> tag or external CSS file** is encouraged in order to maintain a distinction between **content and presentation**.
- Inline styles override any CSS in a <style> tag or external style sheet. this fact more often than not reduces a project's maintainability.
- The styles in the following example apply directly to the elements to which they are attached.
- Inline styles are generally the safest way to ensure rendering compatibility across various email clients, programs and devices, but can be time-consuming to write and a bit challenging to manage.

  Example:

  <h1 style="color: green; text-decoration: underline;">Hello world!</h1>

  ** I ♥ CSS **

CSS TEXT:

1.Text Color

The color property is used **to set the color of the text**. The color is specified by:

- **a color name - like "red"**
- **a HEX value - like "#ff0000"**
- **an RGB value - like "rgb(255,0,0)"**

The default text color for a page is defined in the body selector.

Example:

```
body {
  color: blue;
}

h1 {
  color: green;
}
```

2.Text Alignment

The `text-align` property is used **to set the horizontal alignment of a text**.

A text can be **left or right aligned, centered, or justified**.

- The left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left.

  Example:

```
h1 {
    text-align: center;
   }

h2 {
   text-align: left;
   }

h3 {
    text-align: right;
  }
```

- When the `text-align` property is set to **"justify**", each line is stretched so that every line **has equal width**, and the left and right margins are **straigh**t (like in magazines and newspapers):

Example:

```
div {
  text-align: justify;
}
```

3.Text Decoration

- The `text-decoration` property is used **to set or remove decorations from** text.
- The value `text-decoration: none;` is often used to remove underlines from links:

Example: for anchor tag

```
a {
text-decoration: none;
  }
```

The other `text-decoration` values are used to decorate text:

```
h1 {
text-decoration: overline;
    }

 h2 {
text-decoration: line-through;
    }

 h3 {
text-decoration: underline;
    }
```

4.Text Transformation

- The `text-transform` property is used to **specify uppercase and lowercase** letters in a text.
- It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Example:

```
p.uppercase {
  text-transform: uppercase;
```

```
}

p.lowercase {
  text-transform: lowercase;}

p.capitalize {
  text-transform: capitalize;
}
```

Here the p is the paragraph tag. Among multiple paragraphs we use the class attribute to specify .it is written within the body tag.

```
<p class="uppercase">This is some text.</p>

<p class="lowercase">This is some text.</p>

<p class="capitalize">This is some text.</p>
```

5.Text Indentation

The `text-indent` property is used to **specify the indentation of the first line** of a text:

Example:

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

6.Letter Spacing

The `letter-spacing` property is used to **specify the space between the characters in a text.**

The following example demonstrates how to increase or decrease the space between characters.

Example:

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

h2 {
```

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

7.Line Height

The `line-height` property is used to **specify the space between lines.**

Example:

```
 p.small {
         line-height: 0.8;
        }

p.big {
        line-height: 1.8;
      }
```

8.Text Direction

The `direction` property is used to **change the text direction of an element**.

Example:

```
p {
   direction: rtl;
 }
```

9.Word Spacing

The `word-spacing` property is used to specify **the space between the words in a text.**

The following example demonstrates how to increase or decrease the space between words:

Example:

```
h1 {
    word-spacing: 10px;
    }

 h2 {
```

```
word-spacing: -5px;
}
```

10.Text Shadow

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;
   }
```

Property	Description
color	Sets the color of text
direction	Specifies the text direction/writing direction
letter-spacing	Increases or decreases the space between characters in a text
line-height	Sets the line height
text-align	Specifies the horizontal alignment of text
text-decoration	Specifies the decoration added to text
text-indent	Specifies the indentation of the first line in a text-block
text-shadow	Specifies the shadow effect added to text
text-transform	Controls the capitalization of text
text-overflow	Specifies how overflowed content that is not displayed should be signaled to the user
unicode-bidi	Used together with the direction property to set or return whether the text should be overridden to support multiple languages in the same document
vertical-align	Sets the vertical alignment of an element
white-space	Specifies how white-space inside an element is handled
word-spacing	Increases or decreases the space between words in a text

CSS FONTS

CSS Font Families

In CSS, there **are two types of font family names**:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

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
- **Note**: If the name of a font family is **more than one word, it must be in quotation marks,** like: "Times New Roman".
- More than one font family is specified in a comma-separated list.

Example:

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

Font Style

The `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)

example:

```
p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
```

Font Size

- The `font-size` property **sets the size of the text**.

- Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.
- The font-size value can be an **absolute, or relative size**.

Absolute size:

- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, **the default size for normal text, like paragraphs, is 16px**.

Set Font Size With Pixels

Example

```
h1 {
      font-size: 40px;
   }

h2 {
      font-size: 30px;
   }


 p {
    font-size: 14px;
    }
```

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 */
}

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
}

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
```

Use a Combination of Percent and Em

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

Example

```
body {
  font-size: 100%;
}

h1 {
  font-size: 2.5em;
}

h2 {
  font-size: 1.875em;
}

p {
  font-size: 0.875em;
}
```

CSS SELECTORS

Introduction to selectors:

- CSS selectors **identify specific HTML elements as targets for CSS styles.**
- Selectors use a wide range of over 50 selection methods offered by the **CSS language, including elements, classes, IDs, pseudo-elements and pseudo-classes, and patterns.**
- CSS selectors form **the basis of Cascading Style Sheets, allowing us to target specific elements within a HTML document and apply style.**

List of CSS Selectors:

1. **universal selector (*)**
2. **Type selector / element selector**
3. **Class Selector (.)**
4. **ID selector (#)**
5. **Attribute selector**
6. **PseudoClasses**

1.universal selector: (*)

- **CSS universal selectors** select any type of elements in an HTML page. It matches a single element.
- An **asterisk (i.e. "*")** is used to denote a **CSS universal selector**. An asterisk can also be followed by a selector.
- This is useful when you want to set a style for of all the elements of an HTML page or for all of the elements within an element of an HTML page.

Syntax:

```
synatx:

* { property:value }
```

Example:

```
* {
    color: blue;
  background: silver;
 }
```

2.Type selectors:

Type Selector is also called as Element selector.

- The CSS element Selector or the type selector matches occurrences of those tags specified in the list.
- **Type selectors target element types on the page.** They're also called **element selectors because we use the element's HTML tag as the selector.**

```
synatx:

element { property:value }
```

e.g. `p { font-size:32px; }`

3.class selectors: (.)

The class name selector select all elements with the targeted class name. For example, the **class name. warning** would select the following element:

```
<div class="warning">

<p>This would be some warning copy.</p>

 </div>
```

We can **combine class names to target elements** more specifically. Let's build on the example above to showcase a more complicated class selection.

CSS

```
.important {   color: orange; }

.warning {   color: blue; }

.warning.important {    color: red; }
```

HTML

```
<div class="warning">

 <p>This would be some warning copy</p>

</div>

<div class="important warning">

  <p class="important">This is some really important warning copy</p>

</div>
```

4.ID selector: (#)

ID selectors **select DOM elements with the targeted ID**. To select an element by a specific ID in CSS, **the # prefix** is used.

Example:

```
<div id="hello">

   <p>Example</p>

</div>
```

In css:

```
#hello {

 width: 20px;

}
```

Note: the id selector is a unique. And only one tag can be called at a time.

5.Attribute selector:

The CSS **attribute selector** matches elements based on the presence or value of a given attribute. **(OR)**

- The `[attribute]` selector is used to select elements with a specified attribute.

Example:

```
a[target] {
  background-color: yellow;
       }
```

1.[attribute="value"] Selector

- The `[attribute="value"]` selector is used to select elements with a specified attribute and value.

The following example selects all <a> elements with a **target="_blank"** attribute:

Example:

```
  a[target="_blank"] {
background-color: yellow;

    }
```

2.[attribute~="value"] Selector

The `[attribute~="value"]` selector is used to select elements with an attribute value containing a specified word.

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Example:

```
[title~="flower"] {
       border: 5px solid yellow;
                  }
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

3.[attribute|="value"] Selector

The `[attribute|="value"]` selector is used to select elements with the specified attribute starting with the specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text"!

Example:

```
[class|="top"] {
             background: yellow;
               }
```

4.[attribute^="value"] Selector

The `[attribute^="value"]` selector is used to select elements whose attribute value begins with a specified value.

The following example selects all elements with a class attribute value that begins with "top":

Note: The value does not have to be a whole word!

Example:

```
[class^="top"] {
  background: yellow;
              }
```

5.[attribute$="value"] Selector

The `[attribute$="value"]` selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

Example:

```
[class$="test"] {
     background: yellow;
         }
```

6.[attribute*="value"] Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

Example:

```
[class*="te"] {
       background: yellow;
             }
```

Pseudo Classes:

- Pseudo classes are keywords which allow selection based on information that lies outside of the document tree or that cannot be expressed by other selectors or combinators.
- This information can be associated to a certain **state (state and dynamic pseudo-classes)**, to **locations**

(structural and target pseudo-classes), to **negations of the former (negation pseudo-class)** or to **languages (lang pseudo-class)**.

- Examples include whether or not a link has been followed (:visited), the mouse is over an element (:hover), a checkbox is checked (:checked), etc.

Syntax:

```
selector:pseudo-class {
property: VALUE;
 }
```

LIST OF PSEUDO CLASSES AND DESCRIPTION:

:active	Applies to any element being activated (i.e. clicked) by the user
:any	Allows you to build sets of related selectors by creating groups that the included items will match. This is an alternative to repeating an entire selector.
:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:checked	Applies to radio, checkbox, or option elements that are checked or toggled into an "on" state
:default	Represents any user interface element that is the default among a group of similar elements
:disabled	Applies to any UI element which is in a disabled state
:empty	Applies to any element which has no children
:enabled	Applies to any UI element which is in an enabled state.
:first	Used in conjunction with the @page rule, this selects the first page in a printed document.
:first-child	Represents any element that is the first child element of its parent.
:first-of-type	Applies when an element is the first of the selected element type inside its parent. This may or may not be the first-child.
:focus	Applies to any element which has the user's focus. This can be given by the user's keyboard, mouse events, or other forms of input.
:full-screen	Applies to any element displayed in full-screen mode. It selects the whole stack of elements and not just the top level element.

:hover	Applies to any element being hovered by the user's pointing device, but not activated.
:indeterminate	Applies radio or checkbox UI elements which are neither checked nor unchecked, but are in an indeterminate state. This can be due to an element's attribute or DOM manipulation.
:in-range	CSS pseudo-class matches when an element has its value attribute inside the specified range limitations for this element. It allows the page to give a feedback that the value currently defined using the element is inside the range limits
:invalid	Applies to <input> elements whose values are invalid according to the type specified in the type= attribute.
:left	Used in conjunction with the @page rule, this selects all the left pages in a printed document
:link	Applies to any links which haven't been visited by the user.
:lang	Applies to any element who's wrapping <body> element has a properly designated lang= attribute. For the pseudo-class to be valid, it must contain a valid two or three letter language code.
:last-child	Represents any element that is the last child element of its parent.
:last-of-type	Applies when an element is the last of the selected element type inside its parent. This may or may not be the last-child.
:not()	Applies to all elements which do not match the value passed to **(:not(p) or :not(.class-name)** for example. It must have a value to be valid and

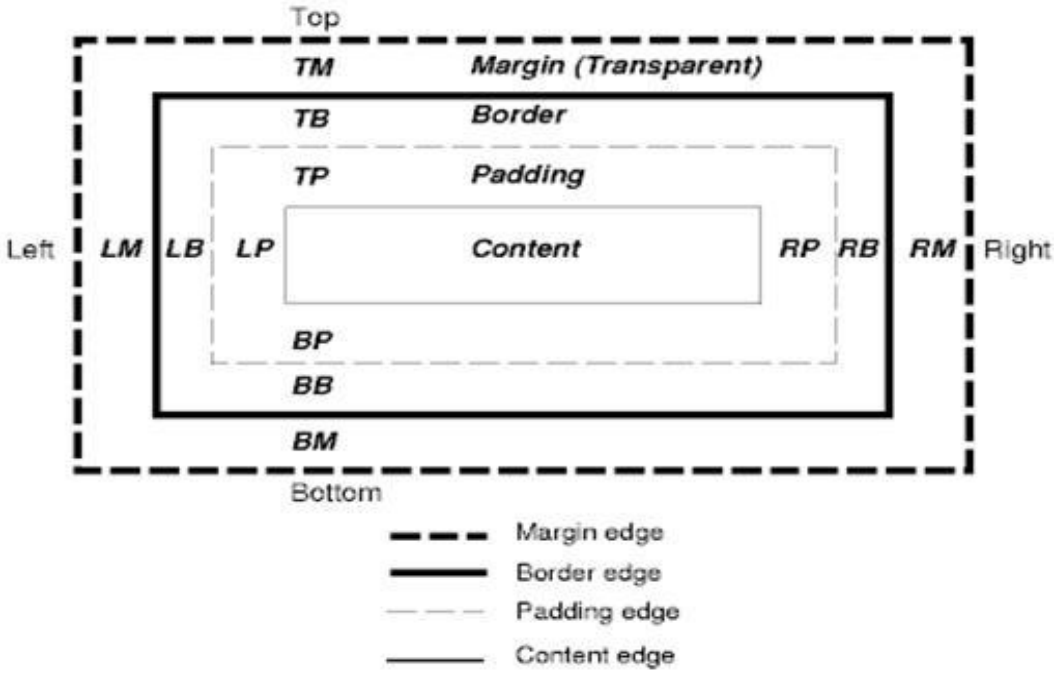
	it can only contain one selector. However, you can chain multiple :not selectors together.
:nth-child	Applies when an element is the n-th element of its parent, where n can be an integer, a mathematical expression (e.g n+3) or the keywords odd or even.
:nth-of-type	Applies when an element is the n-th element of its parent of the same element type, where n can be an integer, a mathematical expression (e.g n+3) or the keywords odd or even.
:only-child	The :only-child CSS pseudo-class represents any element which is the only child of its parent. This is the same as :first-child :last-child or :nth-child(1):nth-last-child(1), but with a lower specificity.
:optional	The:optional CSS pseudo-class represents any element that does not have the required attribute set on it. This allows forms to easily indicate optional fields and to style them accordingly. :out-of-range
The :out-of-range	CSS pseudo-class matches when an element has its value attribute outside the specified range limitations for this element. It allows the page to give a feedback that the value currently defined using the element is outside the range limits. A value can be outside of a range if it is either smaller or larger than maximum and minimum set values.

:placeholder-shown Experimental	Applies to any form element currently displaying placeholder text.
:read-only	Applies to any element which is not editable by the user.
:read-write	Applies to any element that is editable by a user, such as <input> elements.
:right	Used in conjunction with the @page rule, this selects all the right pages in a printed document
:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:visited	Pseudo class can't be used for most styling in a lot of modern browsers anymore because it's a security hole.
:visited	Applies to any links which have has been visited by the user.
:root	matches the root element of a tree representing the document.
:scope	CSS pseudo-class matches the elements that are a reference point for selectors to match against

LAYOUT

BOM(BROWSER OBJECT MODEL / BOX MODEL)

The Edges



- The browser creates a rectangle for each element in the HTML document.
- The Box Model describes how the **padding, border, and margin** are added to the content to create this rectangle.
- The perimeter of each of the four areas is called an **edge**. **Each edge defines a box**.
- The innermost rectangle is the **content box**. The width and height of this depends on the element's rendered content (text, images and any child elements it may have).

- Next is the **padding box**, as defined by the padding property. If there is no padding width defined, the padding edge is equal to the content edge.
- the **border box**, as defined by the border property. If there is no border width defined, the border edge is equal to the padding edge.
- The outermost rectangle is the **margin box**, as defined by the margin property. If there is no margin width defined, the margin edge is equal to the border edge.

Example

```
div {
border: 5px solid red;
margin: 50px;
padding: 20px;
}
```

box-sizing

The default box model (content-box) can be counter-intuitive, since the width / height for an element will not represent its actual width or height on screen as soon as you start adding padding and border styles to the

The following example demonstrates this potential issue with content-box:

Eg:

```
textarea {
    width: 100%;
```

```
    padding: 3px;

    box-sizing: content-box; /* default value */ }
```

POSITIONING

Types of positioning section :

- A **positioned element** is an element whose computed position value is either **relative, absolute, fixed, or sticky**. (In other words, it's anything except static.)
- A **relatively positioned element** is an element whose computed position value is **relative**.
  The top and bottom properties specify the vertical offset from its normal position; the left and right properties specify the horizontal offset.
- An **absolutely positioned element** is an element whose computed position value is **absolute or fixed**.
  The top, right, bottom, and left properties specify offsets from the edges of the element's containing block. (The containing block is the ancestor relative to which the element is positioned.) If the element has margins, they are added to the offset.
- A **stickily positioned element** is an element whose computed position value **is sticky**. It's treated as relatively positioned until its containing block crosses a specified threshold (such as setting top to value other than auto) within its flow root (or the container it scrolls within), at which point it is treated as "stuck" until meeting the opposite edge of its containing block.

Backgrounds

With CSS you can set **colors, gradients, and images as the background of an element.**
It is possible to specify various combinations of **images, colors, and gradients, and adjust the size, positioning, and repetition** (among others) of these.

Background Color :

The background-color property sets the background color of an element using a color value or through keywords, such **as transparent, inherit or initial**.

- **transparent**, specifies that the background color should be transparent. This is default.
- **inherit,** inherits this property from its parent element.
- **initial**, sets this property to its default value.

This can be applied to **all elements, and ::first-letter/::first-line pseudo-elements.**

Colors in CSS can be specified by different methods.

1.Color names

CSS

```
div {
background-color: red;  /* red */
}
```

HTML

```
<div>This will have a red background</div>
```

2.Hex color codes

Hex code is used **to denote RGB components** of a color in **base-16 hexadecimal notation**.

#ff0000, for example, is bright red, where the red component of the color is 256 bits (ff) and the corresponding green and blue portions of the color is 0 (00).

Note: If both values in each of the three RGB pairings (R, G, and B) are the same, then the color code can be shortened into three characters (the first digit of each pairing).

#ff0000 can be shortened to #f00, and

#ffffff can be shortened to #fff.

Note: Hex notation is case-insensitive.

Html:

```
body {
  background-color: #de1205; /* red */
}
```

Css:

```
.main {
  background-color: #00f; /* blue */
}
```

RGB / RGBA

Another way to declare a color is to use RGB or RGBA.

RGB stands for **Red, Green and Blue,** and requires of three separate values between **0 and 255**, put between brackets, that correspond with the decimal color values for respectively red, green and blue.

RGBA allows you to add an additional alpha parameter **between 0.0 and 1.0 to define opacity.**

Html:

```
header {
 background-color: rgb(0, 0, 0); /* black */
 }
```

```
footer {
-color: rgba(0, 0, 0, 0.5); /* black with 50% opacity */
}
```

HSL / HSLA

Another way to declare a color is to use HSL or HSLA and is similar to RGB and RGBA.

HSL stands for **hue, saturation, and lightness**, and is also often called HLS:

- ✓ **Hue** is a degree on the color wheel **(from 0 to 360).**
- ✓ **Saturation** is a percentage between **0% and 100%.**
- ✓ **Lightness** is also a percentage between **0% and 100%.**

HSLA allows you to add an additional alpha parameter **between 0.0 and 1.0 to define opacity.**

```
li a {
 background-color: hsl(120, 100%, 50%); /* green */
 }
```

Css:

```
#p1 {
 background-color: hsla(120, 100%, 50%, .3); /* green with 30% opacity */
 }
```

Interaction with background-image

The following statements are all equivalent:

```
body {  background: red;
background-image: url(partiallytransparentimage.png);
}
body {  background-color: red;
background-image: url(partiallytransparentimage.png);
}
body {  background-image: url(partiallytransparentimage.png);
```

```
background-color: red;
}
body {
background: red url(partiallytransparentimage.png);
 }
```

They will all lead to the red color being shown underneath the image, where the parts of the **image are transparent,** or the image is not showing (perhaps as a result of background-repeat).

Note that the following is not equivalent:

```
body {  background-image: url(partiallytransparentimage.png);
background: red;
}
```

Here, the value of **background overrides your background-image.**

Background Gradients

Gradients are **new image types,** added in **CSS3**. As an image, **gradients are set with the background-image property**.

There are **two types of gradient functions**, **linear and radial**. Each type has a **non-repeating variant and a repeating variant**:

1.linear-gradient()

2.repeating-linear-gradient()

3.radial-gradient()

4.repeating-radial-gradient()

1.linear-gradient()

A linear-gradient has the following syntax

background: linear-gradient(<direction>?, <color-stop-1>, <color-stop-2>, …); Value Meaning

<direction>

Could be an argument like to **top, to bottom, to right or to left**; or an **angle as 0deg, 90deg**... . The angle starts from to **top and rotates clockwise**. Can be specified in deg, grad, rad, or turn. If omitted, the gradient flows from top to bottom

<color-stop-list> List of colors, optionally followed each one by a percentage or length to display it at.
For example,
yellow 10%, rgba(0,0,0,.5) 40px, #fff 100%...

For example,
this creates a **linear gradient** that starts from the right and transitions from red to blue

```
.linear-gradient {  background: linear-gradient(to left, red, blue); /* you can also use 270deg */ }
```

You can create **a diagonal gradient** by declaring both a **horizontal and vertical starting position.**

```
.diagonal-linear-gradient {  background: linear-gradient(to left top, red, yellow 10%);
}
```

The following examples will **create a gradient with 8 color stops**

```
.linear-gradient-rainbow {  background: linear-gradient(to left, red, orange, yellow, green, blue, indigo, violet) }
```

radial-gradient() .radial-gradient-simple { background: radial-gradient(red, blue); }
.radial-gradient { background: radial-gradient(circle farthest-corner at top left, red, blue); }

- ✓ **Value Meaning** circle Shape of gradient. Values are circle or ellipse, **default is ellipse**.
- ✓ **farthest-corner** Keywords describing how big the ending shape must be.
- ✓ Values are **closest-side, farthestside, closest-corner, farthest-corner** top left Sets the position of the gradient center, in the same way as background-position.

Repeating gradients

Repeating gradient functions take the same arguments as the above examples, but **tile the gradient across the background of the element**.
.bullseye { background: repeating-radial-gradient(red, red 10%, white 10%, white 20%); }
.warning { background: repeating-linear-gradient(-45deg, yellow, yellow 10%, black 10%, black 20%); } Value Meaning - 45deg Angle unit.

The angle starts from to top and rotates clockwise. Can be specified **in deg, grad, rad, or turn. to left Direction of gradient**, **default is to bottom**.

Syntax:

to [y-axis(top OR bottom)] [x-axis(left OR right)] ie to top right yellow 10% Color, optionally followed by a percentage or length to display it at. Repeated two or more times.

Note

1. that HEX, RGB, RGBA, HSL, and HSLA color codes may be used instead of color names. Color names were used for the sake of illustration.

2.Also note that the radial-gradient syntax is much more complex than linear-gradient,

Background Image

The background-image property is used to specify a background image to be applied to all matched elements. By default, this image is tiled to cover the entire element, excluding margin.

```
.myClass {
background-image: url('/path/to/image.jpg');
}
```

To use multiple images as background-image, define comma separated url()

```
.myClass {
 background-image: url('/path/to/image.jpg'),      url('/path/to/image2.jpg');
}
```

- ✓ The images will stack according to their order with the first declared image on top of the others and so on.

- ✓ Value Result url('/path/to/image.jpg') Specify background image's path(s) or an image resource specified with data URI schema (apostrophes can be omitted), separate multiples by comma none No background image initial **Default value inherit Inherit parent's value**

CSS Transformation

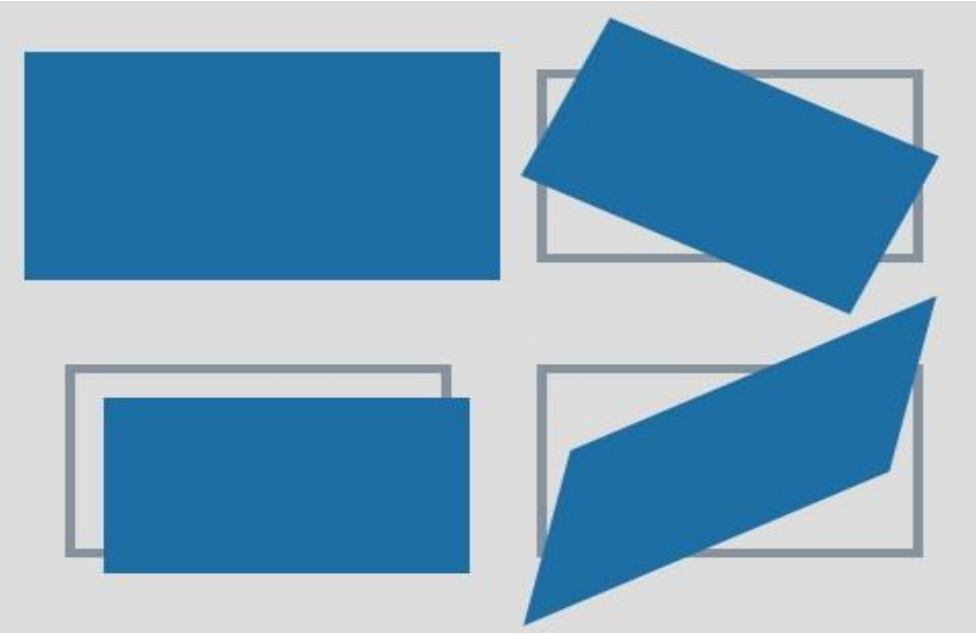
Scale, Rotate, Translate and Skew

Scale works like you would zoom in and out the targeted element. The default *scale* value is 1, which works as a multiplier of the original size. This means that 0.5 halves while 2 doubles the section.

Rotate the element clockwise with the second property that's set in degrees. Turning with 180° puts the object upside down while 360° takes is back to its original upright position. Set any positive or negative value or even decimals.

Translate shifts the element with pixels related to its original position. The X value horizontally while Y vertically when there rotate attribute is zero.

Skew the objects on their horizontal (X) or vertical (Y) axle.



Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.a {
  width: 150px;
  height: 80px;
  background-color: yellow;
  -ms-transform: rotate(20deg); /* IE 9 */
  -webkit-transform: rotate(20deg); /* Safari 3-8 */
  transform: rotate(20deg);
}
div.b {
  width: 150px;
  height: 80px;
  background-color: yellow;
  -ms-transform: skewY(20deg); /* IE 9 */
  -webkit-transform: skewY(20deg); /* Safari 3-8 */
  transform: skewY(20deg);
}
div.c {
  width: 150px;
  height: 80px;
  background-color: yellow;
  -ms-transform: scaleY(1.5); /* IE 9 */
  -webkit-transform: scaleY(1.5); /* Safari 3-8 */
  transform: scaleY(1.5);
}
</style>
</head>
```

```
<body>
<h1>The transform Property</h1>
<h2>transform: rotate(20deg):</h2>
<div class="a">Hello World!</div>
<br>

<h2>transform: skewY(20deg):</h2>
<div class="b">Hello World!</div>
<br>

<h2>transform: scaleY(1.5):</h2>
<div class="c">Hello World!</div>

</body>
</html>
```