

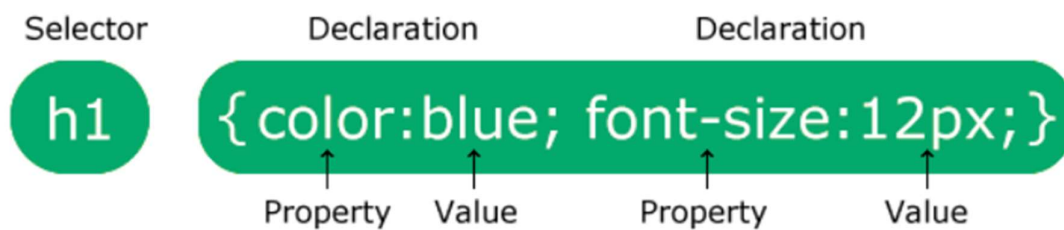
# Introduction to CSS

## What is CSS?

### Def

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

### Syntax



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Declaration blocks are surrounded by curly braces.

### Example

```
h1 {  
  color: white;  
  text-align: center;  
}  
  
p {  
  font-family: verdana;  
  font-size: 20px;  
}
```

### Css Solved a Big problem

- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers.
- Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- CSS removed the style formatting from the HTML page!

## Four ways to insert CSS

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS
- Importing CSS

### 1. External Css

- Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.
- The external .css file should not contain any HTML tags.

```
<link rel="stylesheet" href="mystyle.css">
```

### 2. Internal Css

- The internal style is defined inside the <style> element, inside the head section.
- An internal style sheet may be used if one single HTML page has a unique style.

```
<head> <style>
h1 {
  color: maroon;
  margin-left: 40px;
}
</style> </head>
```

### 3. Inline Css

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element.

```
<h1 style="color:blue;text-align:center;">This is a heading</h1>
```

### 4. Importing Css

```
<style type="text/css">
```

```
  @import url("http://www.xyz.com/style.css")
```

```
</style>
```

## Cascading Order

### 1. Order of style sheet

All the styles in a page will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. Internal style (in the head section) or Import url
3. External style sheet
4. Browser default

## 2. Selector Precedence

If there are multiple styles specified for the same HTML element then the priority is as below.

1. Id
2. Class
3. Element

## 3. !important keyword

In case if any style property is important, and should not be overwritten by any, other selectors, we can specify the !important keyword against the property.

*color: limegreen !important;*

## 4. Specificity Concept

When there are multiple styles specified using multiple selectors in a style rule, then the priority is to be decided based on the specificity

A Specificity is a 3-digit number.

if the specificity of a style rule is 123 then,

- 1 refers to the number of id selectors
- 2 refers to the number of class selectors
- 3 refers to the number of elements

The style rule with the highest specificity always wins.

```
<section class="sclass" id="sid">
  <p class="pclass" id="pid">Cascading order</p>
</section>

#sid #pid {
  color: blue;
}

section.sclass p#pid.pclass {
  color: green;
}
```

The specificity of the first statement is 200 (2 id selectors, 0 class selectors, and 0 element selectors)

The specificity of the second statement is 122 (1 id selector, 2 class selectors, and 2 element selectors)

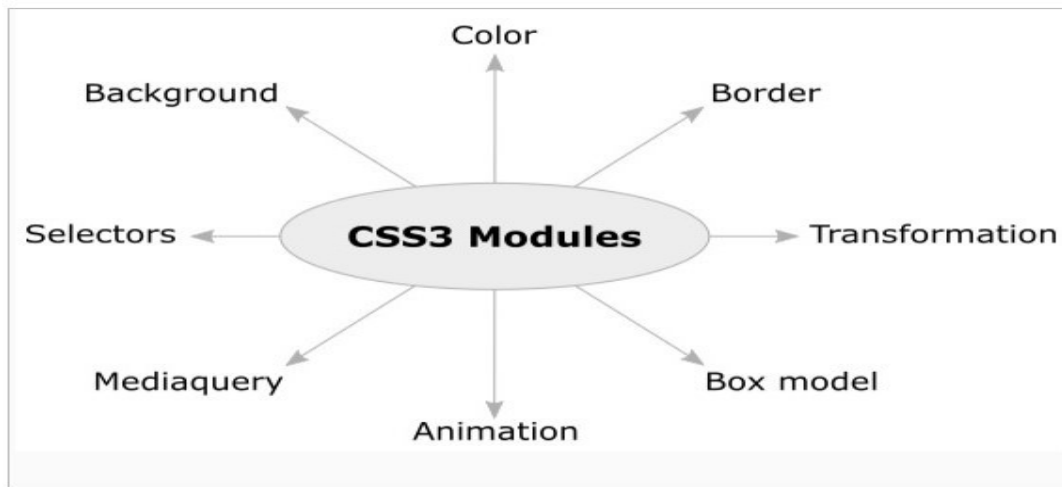
The highest specificity (200) wins so the first styling statement will be prioritized.

## Comments

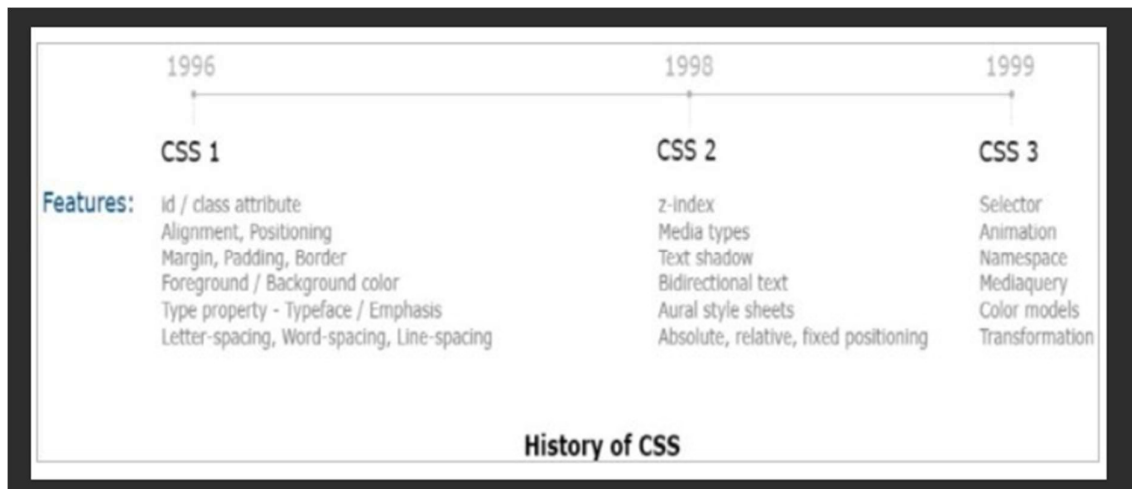
A CSS comment is placed inside the <style> element, and starts with /\* and ends with \*/:

*/\* This is a single-line comment \*/*

## CSS Modules



## History of CSS



# CSS Background

## Def

The CSS background properties are used to add background effects for elements.

Some of the background properties are:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background (shorthand property)

## 1. Background-Color

### Def

```
body {  
  background-color: lightblue;  
}
```

### Opacity / Transparency

It can take a value from 0.0 - 1.0.

The lower value, the more transparent.

When using the opacity, all of its child elements inherit the same transparency.

### Transparency using RGBA

If you do not want to apply opacity to child elements, use **RGBA** color values.

An RGBA color value is specified with: `rgba(red, green, blue, alpha)`.

The *alpha* parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
div {  
  background: rgba(0, 128, 0, 0.3) /* Green background with 30% opacity */  
}
```

## 2. Background image

The background-image property specifies an image to use as the background of an element.

```
body {  
  background-image: url("paper.gif");  
}
```

## 3. Background repeat

By default, the background-image property repeats an image both horizontally and vertically.

If the image above is repeated only horizontally (`background-repeat: repeat-x;`), the background will look better

To repeat an image vertically, set `background-repeat: repeat-y;`

```
body {  
  background-image: url("gradient_bg.png");  
  background-repeat: repeat-x;  
}  
  
background-repeat: no-repeat;
```

#### 4. **Background position**

Sets the starting position of a background image

```
body {  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
}
```

#### 5. **Background attachment**

Sets whether a background image is fixed or scrolls with the rest of the page

```
body {  
  background-image: url("img_tree.png");  
  background-attachment: scroll;  
}
```

#### 6. **Background Shorthand Property**

To shorten the code, it is also possible to specify all the background properties in one single property. This is called a shorthand property.

Instead of writing:

```
body {  
  background-color: #ffffff;  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
}
```

You can use the shorthand property background:

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

| Property                          | Description                                                                       |
|-----------------------------------|-----------------------------------------------------------------------------------|
| <a href="#">background</a>        | A shorthand property for setting all the background properties in one declaration |
| <a href="#">background-clip</a>   | Specifies the painting area of the background                                     |
| <a href="#">background-image</a>  | Specifies one or more background images for an element                            |
| <a href="#">background-origin</a> | Specifies where the background image(s) is/are positioned                         |
| <a href="#">background-size</a>   | Specifies the size of the background image(s)                                     |

# CSS Borders

## Def

The CSS border properties allow you to specify the style, width, and color of an element's border.

### 1. 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.
- ridge - Defines a 3D ridged border
- inset - Defines a 3D inset border.
- outset - Defines a 3D outset border.
- none - Defines no border
- hidden - Defines a hidden border

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

### 2. Border width

The 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:

```
p.one {  
  border-style: solid;  
  border-width: 5px;  
}  
  
p.one {  
  border-style: solid;  
  border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */  
}
```

### 3. Border Color

The 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)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

```
p.one {
  border-style: solid;
  border-width: 5px;
}

p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */
}
```

#### 4. Border Sides

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

#### 5. Border radius

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

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

#### 6. Border Shorthand

The border property is a shorthand property for the following individual border properties:

- border-width
- border-style (required)
- border-color

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

#### Border Tags

- |                        |                        |                        |
|------------------------|------------------------|------------------------|
| 1. border              | 7. border-left         | 15. border-right-width |
| 2. border-bottom       | 8. border-left-color   | 16. border-style       |
| 3. border-bottom-      | 9. border-left-style   | 17. border-top         |
| color                  | 10. border-left-width  | 18. border-top-color   |
| 4. border-bottom-style | 11. border-radius      | 19. border-top-style   |
| 5. border-bottom-      | 12. border-right       | 20. border-top-width   |
| width                  | 13. border-right-color | 21. border-width       |
| 6. border-color        | 14. border-right-style |                        |



# CSS Margin

## Def

The CSS 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

All the margin properties can have the following values:

- auto - the browser calculates the margin
- *length* - specifies a margin in px, pt, cm, etc.
- % - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

## Margin Shorthand Property

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

## Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

```
h1 {  
  margin: 0 0 50px 0;  
}
```

```
h2 {  
  margin: 20px 0 0 0;  
}
```

# CSS Padding

## Def

The 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

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

## Padding Shorthand Property

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

## Padding and Element width

```
div {  
  width: 300px;  
  padding: 25px;  
}
```

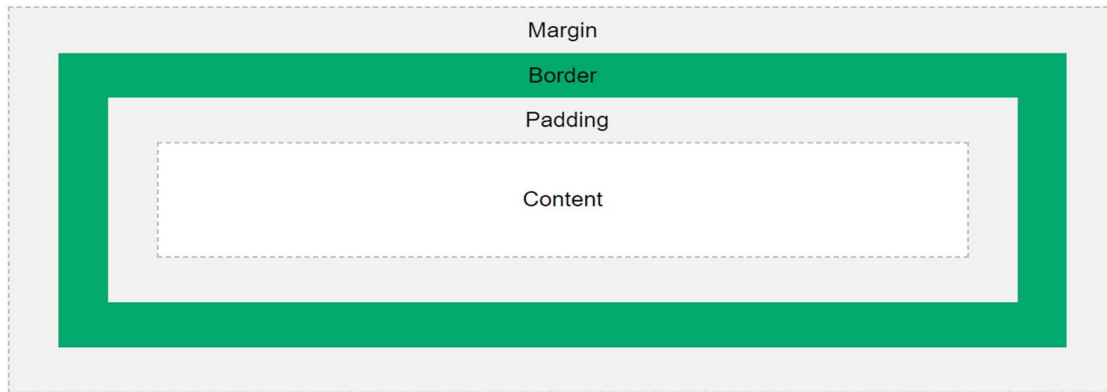
This div is 300px wide.

The width of this div is 350px, even though it is defined as 300px in the CSS.

# CSS Box Model

## Def

- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element.
- It consists of: content, padding, borders and margins.



- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

## Total Width and Total height calculation

```
div {  
  width: 320px;  
  height: 50px;  
  padding: 10px;  
  border: 5px solid gray;  
  margin: 0;  
}
```

320px (width of content area)  
+ 20px (left padding + right padding)  
+ 10px (left border + right border)  
**= 350px (total width)**

50px (height of content area)  
+ 20px (top padding + bottom padding)  
+ 10px (top border + bottom border)  
**= 80px (total height)**

# CSS Outline

## Def

An outline is a line drawn outside the element's border.



CSS has the following outline properties:

- outline-style
- outline-color
- outline-width
- outline-offset
- outline

## Outline Shorthand

The outline property is a shorthand property for setting the following individual outline properties:

- outline-width
- outline-style (required)
- outline-color

## Syntax

*p.ex4 {outline: thick ridge pink;}*

## Outline Offset

The outline-offset property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

This paragraph has an outline 15px outside the border edge.

# CSS Typography

## Def

CSS has a lot of properties for formatting text.

## 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)"

## Text Alignment and Text Direction

[https://www.w3schools.com/css/css\\_align.asp](https://www.w3schools.com/css/css_align.asp)

| Property                        | Description                                                                                                                                                         |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <a href="#">direction</a>       | Specifies the text direction/writing direction                                                                                                                      |
| <a href="#">text-align</a>      | Specifies the horizontal alignment of text                                                                                                                          |
| <a href="#">text-align-last</a> | Specifies how to align the last line of a text                                                                                                                      |
| <a href="#">unicode-bidi</a>    | Used together with the <a href="#">direction</a> property to set or return whether the text should be overridden to support multiple languages in the same document |
| <a href="#">vertical-align</a>  | Sets the vertical alignment of an element                                                                                                                           |

## Text Decoration

| Property                                  | Description                                                                  |
|-------------------------------------------|------------------------------------------------------------------------------|
| <a href="#">text-decoration</a>           | Sets all the text-decoration properties in one declaration                   |
| <a href="#">text-decoration-color</a>     | Specifies the color of the text-decoration                                   |
| <a href="#">text-decoration-line</a>      | Specifies the kind of text decoration to be used (underline, overline, etc.) |
| <a href="#">text-decoration-style</a>     | Specifies the style of the text decoration (solid, dotted, etc.)             |
| <a href="#">text-decoration-thickness</a> | Specifies the thickness of the text decoration line                          |

## Text Transform

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.

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

## Text Spacing

| Property                    | Description                                                 |
|-----------------------------|-------------------------------------------------------------|
| <code>letter-spacing</code> | Specifies the space between characters in a text            |
| <code>line-height</code>    | Specifies the line height                                   |
| <code>text-indent</code>    | Specifies the indentation of the first line in a text-block |
| <code>white-space</code>    | Specifies how to handle white-space inside an element       |
| <code>word-spacing</code>   | Specifies the space between words in a text                 |

## Text Shadow

Text shadow specifies the shadow for the text.

```
h1 {  
  text-shadow: 2px 2px 5px red;  
}
```

Where, 2px for Horizontal , 2px for vertical , 5px for blur and red for text color.

It is possible to specify multiple shadows for single element as below,

```
h1 {  
  color: white;  
  text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;  
}
```

## Text Effects

| Property                   | Description                                                                           |
|----------------------------|---------------------------------------------------------------------------------------|
| <code>text-justify</code>  | Specifies how justified text should be aligned and spaced                             |
| <code>text-overflow</code> | Specifies how overflowed content that is not displayed should be signaled to the user |
| <code>word-break</code>    | Specifies line breaking rules for non-CJK scripts                                     |
| <code>word-wrap</code>     | Allows long words to be able to be broken and wrap onto the next line                 |
| <code>writing-mode</code>  | Specifies whether lines of text are laid out horizontally or vertically               |

# CSS Fonts

## Generic Font Families

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

## Font family

In CSS, we use the font-family property to specify the font of a text.

**Tip:** The font-family property should hold several font names as a "fallback" system, to ensure maximum compatibility.

```
.p1 {  
  font-family: "Times New Roman", Times, serif;  
}
```

## Web safe fonts

Web safe fonts are fonts that are universally installed across all browsers and devices.

## Fallback fonts

However, there are no 100% completely web safe fonts.

There is always a chance that a font is not found or is not installed properly.

Therefore, it is very important to always use fallback fonts.

This means that you should add a list of similar "backup fonts" in the font-family property.

If the first font does not work, the browser will try the next one, and the next one, and so on. Always end the list with a generic font family name.

## Font style

1. font-style
2. font-weight
3. font-variant

## Font Size

1. Set Font Size With Pixels
2. Set Font Size With Em ( 1em = 16 px)
3. Use a Combination of Percent and Em
4. Responsive Font Size → vw (viewport width)

## Google Fonts and Effects

Google Fonts are free to use, and have more than 1000 fonts to choose from.

Just add a special style sheet link in the <head> section and then refer to the font in the CSS.

```
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
```

To use multiple Google fonts, just separate the font names with a pipe character (|).

First add `effect=effectname` to the Google API, then add a special class name to the element that is going to use the special effect. The class name always starts with `font-effect-` and ends with the *effectname*.

## Font Pairing rules

1. Use Complements
2. Use Font Superfamilies
3. Contrast is King
4. Choose Only One Boss

**Eg.**

Georgia and Verdana

Helvetica and Garamond

## Fonts Shorthand Property

The font property is a shorthand property for:

- font-style
- font-variant
- font-weight
- font-size/line-height
- font-family

**Note:** The font-size and font-family values are required. If one of the other values is missing, their default value is used.

```
p.a{  
  font: italic small-caps bold 12px/30px Georgia, serif;  
}
```

## CSS Selectors

### Def

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)



- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

## 1. Simple Selectors

A simple selector is either a type selector or universal selector followed immediately by zero or more attribute selectors, ID selectors, or pseudo-classes, in any order.

The simple selector matches if all of its components match.

| Selector                   | Example    | Example description                             |
|----------------------------|------------|-------------------------------------------------|
| <u>#id</u>                 | #firstname | Selects the element with id="firstname"         |
| <u>.class</u>              | .intro     | Selects all elements with class="intro"         |
| <u>element.class</u>       | p.intro    | Selects only <p> elements with class="intro"    |
| <u>*</u>                   | *          | Selects all elements                            |
| <u>element</u>             | p          | Selects all <p> elements                        |
| <u>element,element,...</u> | div, p     | Selects all <div> elements and all <p> elements |

1. Element Selector
2. Class Selector
3. Universal Selector
4. Grouping Selector

## 2. Combinator Selector

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

| Selector                  | Example | Example description                                                            |
|---------------------------|---------|--------------------------------------------------------------------------------|
| <u>element element</u>    | div p   | Selects all <p> elements inside <div> elements                                 |
| <u>element&gt;element</u> | div > p | Selects all <p> elements where the parent is a <div> element                   |
| <u>element+element</u>    | div + p | Selects the first <p> element that are placed immediately after <div> elements |
| <u>element1~element2</u>  | p ~ ul  | Selects every <ul> element that are preceded by a <p> element                  |

## 3. Pseudo Class Selector

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it

- Style visited and unvisited links differently
- Style an element when it gets focus

### Syntax

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

### Other Tags used

- |                          |                     |                     |
|--------------------------|---------------------|---------------------|
| 1. :active               | 5. :enabled         | 9. :hover           |
| 2. :checked              | 6. :first-child     | 10. :in-range       |
| 3. :disabled             | 7. :first-of-type   | 11. :invalid        |
| 4. :empty                | 8. :focus           | 12. :lang(language) |
| 13. :last-child          | 20. :nth-of-type(n) | 26. :required       |
| 14. :last-of-type        | 21. :only-child     | 27. :root           |
| 15. :link                | 22. :optional       | 28. :target         |
| 16. :not(selector)       | 23. :out-of-range   | 29. :valid          |
| 17. :nth-child(n)        | 24. :read-only      | 30. :visited        |
| 18. :nth-last-child(n)   | 25. :read-write     |                     |
| 19. :nth-last-of-type(n) |                     |                     |

## 4. Pseudo Element Selector

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

### Syntax

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

| Selector              | Example         | Example description                                          |
|-----------------------|-----------------|--------------------------------------------------------------|
| <u>::after</u>        | p::after        | Insert content after every <p> element                       |
| <u>::before</u>       | p::before       | Insert content before every <p> element                      |
| <u>::first-letter</u> | p::first-letter | Selects the first letter of every <p> element                |
| <u>::first-line</u>   | p::first-line   | Selects the first line of every <p> element                  |
| <u>::marker</u>       | ::marker        | Selects the markers of list items                            |
| <u>::selection</u>    | p::selection    | Selects the portion of an element that is selected by a user |

## 5. Attribute Selector

| Selector                   | Example              | Example description                                                                                                 |
|----------------------------|----------------------|---------------------------------------------------------------------------------------------------------------------|
| <u>[attribute]</u>         | [target]             | Selects all elements with a target attribute                                                                        |
| <u>[attribute=value]</u>   | [target="_blank"]    | Selects all elements with target="_blank"                                                                           |
| <u>[attribute~value]</u>   | [title~="flower"]    | Selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower" |
| <u>[attribute =value]</u>  | [lang="en"]          | Selects all elements with a lang attribute value starting with "en"                                                 |
| <u>[attribute^=value]</u>  | a[href^="https"]     | Selects all <a> elements with a href attribute value starting with "https"                                          |
| <u>[attribute\$=value]</u> | a[href\$=".pdf"]     | Selects all <a> elements with a href attribute value ending with ".pdf"                                             |
| <u>[attribute*=value]</u>  | a[href*="w3schools"] | Selects all <a> elements with a href attribute value containing the substring "w3schools"                           |

## CSS Links

Links can be styled with any CSS property (e.g. color, font-family, background, etc.). and mainly Text decoration and background color.

links can be styled differently depending on what **state** they are in.

The four links states are:

- a:link - a normal, unvisited link
- a:visited - a link the user has visited
- a:hover - a link when the user mouses over it
- a:active - a link the moment it is clicked

## CSS Lists

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

### **List shorthand property**

```
ul {  
  list-style: square inside url("sqpurple.gif");  
}
```

When using the shorthand property, the order of the property values are:

- list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
- list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
- list-style-image (specifies an image as the list item marker)

If one of the property values above is missing, the default value for the missing property will be inserted, if any.

# CSS Display

## **Def**

The display property is used to specify how an element is shown on a web page.

Every HTML element has a default display value, depending on what type of element it is.

The default display value for most elements is block or inline.

The display property is used to change the default display behavior of HTML elements.

## **Block level Elements**

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).

Examples of block-level elements:

- <div>
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

## **Inline level Element**

An inline element DOES NOT start on a new line and only takes up as much width as necessary.

Examples of inline elements:

- <span>
- <a>
- <img>

## **Display: none vs Visibility: hidden**

Hiding an element can be done by setting the display property to none. The element will be hidden, and the **page will be displayed as if the element is not there**:

visibility:hidden; also hides an element.

However, the element will still take up the same space as before. The **element will be hidden, but still affect the layout**.

## **CSS Height, Width and Max-width**

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

The CSS max-width property is used to set the maximum width of an element.

The height and width properties may have the following values:

- auto - This is default. The browser calculates the height and width

- length - Defines the height/width in px, cm, etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

```
div {
  width: 500px;
  height: 100px;
}
```

```
div {
  max-width: 500px;
  height: 100px;
}
```

## Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the `width` of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to `auto`, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This `<div>` element has a width of 500px, and margin set to `auto`.

**Note:** The problem with the `<div>` above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This `<div>` element has a max-width of 500px, and margin set to `auto`.

**Tip:** Resize the browser window to less than 500px wide, to see the difference between the two divs!

## Block vs inline vs inline-block

### display: inline

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

### display: inline-block

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet

facilisis. Nullam cursus fermentum velit sed laoreet.

### display: block

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat.

Aliquam  
venenatis

gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.


### ***Display property values***

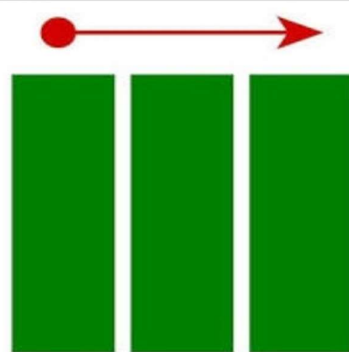
| <b>Value</b>       | <b>Description</b>                                                                                                                                      |
|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| inline             | Displays an element as an inline element                                                                                                                |
| block              | Displays an element as a block element                                                                                                                  |
| contents           | Makes the container disappear, making the child elements children of the element the next level up in the DOM                                           |
| flex               | Displays an element as a block-level flex container                                                                                                     |
| grid               | Displays an element as a block-level grid container                                                                                                     |
| inline-block       | Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values |
| inline-flex        | Displays an element as an inline-level flex container                                                                                                   |
| inline-grid        | Displays an element as an inline-level grid container                                                                                                   |
| inline-table       | The element is displayed as an inline-level table                                                                                                       |
| list-item          | Let the element behave like a <li> element                                                                                                              |
| run-in             | Displays an element as either block or inline, depending on context                                                                                     |
| table              | Let the element behave like a <table> element                                                                                                           |
| table-caption      | Let the element behave like a <caption> element                                                                                                         |
| table-column-group | Let the element behave like a <colgroup> element                                                                                                        |
| table-header-group | Let the element behave like a <thead> element                                                                                                           |

|                    |                                                |
|--------------------|------------------------------------------------|
| table-footer-group | Let the element behave like a <tfoot> element  |
| table-row-group    | Let the element behave like a <tbody> element  |
| table-cell         | Let the element behave like a <td> element     |
| table-column       | Let the element behave like a <col> element    |
| table-row          | Let the element behave like a <tr> element     |
| none               | The element is completely removed              |
| initial            | Sets this property to its default value        |
| inherit            | Inherits this property from its parent element |

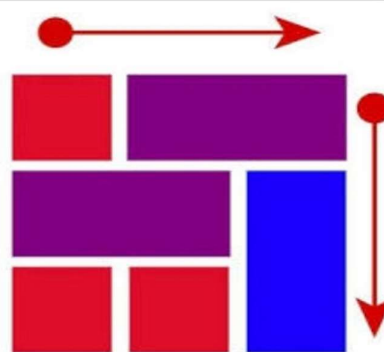


## Flexbox vs Grid

| CSS Flexbox                                                                                        | CSS Grid                                                                                          |
|----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| CSS Flexbox is a one-dimensional layout model.                                                     | CSS Grid is a two-dimensional model.                                                              |
| A Flexbox container can either facilitate laying out things in a row, or lay them out in a column. | Grid can facilitate laying out items across and down at once.                                     |
| Flexbox cannot intentionally overlap elements or items in a layout.                                | CSS Grid helps you create layouts with overlapping elements.                                      |
| Flexbox is basically content based and it listens to the content and adjusts to it.                | Grid operates more on the layout level and it is container based.                                 |
| Flexbox can be used for scaling, one-sided aligning, and organizing elements within a container.   | Grid is useful when you want to define a large-scale layout with more complex and subtle designs. |
|                                                                                                    |              |



Flexbox  
One Dimensions



CSS Grids  
Two Dimensions

# CSS Position

## Def

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

Elements are then positioned using the top, bottom, left, and right properties.

However, these properties will not work unless the position property is set first.

They also work differently depending on the position value.

## 1. static position

An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page.

```
div.static {  
  position: static;  
  border: 3px solid #73AD21;  
}
```

## 2. relative position

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

```
div.relative {  
  position: relative;  
  left: 30px;  
  border: 3px solid #73AD21;  
}
```

## 3. fixed position

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled.

The top, right, bottom, and left properties are used to position the element.

```
div.fixed {  
  position: fixed;  
  bottom: 0;  
  right: 0;  
  width: 300px;  
  border: 3px solid #73AD21;  
}
```

## 4. sticky position

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

```
div.sticky {  
  position: -webkit-sticky; /* Safari */  
  position: sticky;  
  top: 0;  
  border: 2px solid #4CAF50;  
}
```

## 5. absolute position

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

```
div.relative {  
  position: relative;  
  width: 400px;  
  height: 200px;  
  border: 3px solid #73AD21;  
}  
div.absolute {  
  position: absolute;  
  top: 80px;  
  right: 0;  
  width: 200px;  
  height: 100px;  
  border: 3px solid #73AD21;  
}
```

## CSS Z-index

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

When elements are positioned, they can overlap other elements.

An element can have a positive or negative stack order.

## CSS Overflow

### Def

The CSS overflow property controls what happens to content that is too big to fit into an area.

The overflow property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

- visible - Default. The content renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible

- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
- auto - Similar to scroll, but it adds scrollbars only when necessary

### 1. **Overflow: visible**

By default, the overflow is `visible`, meaning that it is not clipped and it renders outside the element's box:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

### 2. **Overflow: hidden**

With the `hidden` value, the overflow is clipped, and the rest of the content is hidden:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what

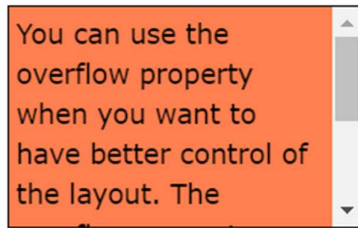
### 3. **Overflow: scroll**

Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

You can use the overflow property when you want to have better control of the layout. The

#### 4. **Overflow: auto**

The `auto` value is similar to `scroll`, but it adds scrollbars only when necessary:

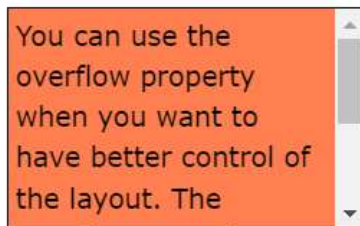


#### 5. **Overflow: x and Overflow: y**

The `overflow-x` and `overflow-y` properties specify whether to change the overflow of content just horizontally or vertically (or both):

`overflow-x` specifies what to do with the left/right edges of the content.

`overflow-y` specifies what to do with the top/bottom edges of the content.



## CSS Float and Clear

### **Float**

The `float` property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The `float` property can have one of the following values:

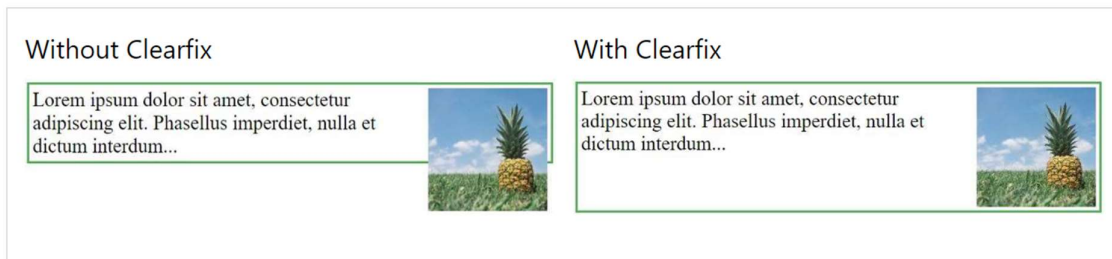
- `left` - The element floats to the left of its container
- `right` - The element floats to the right of its container
- `none` - The element does not float (will be displayed just where it occurs in the text). This is default
- `inherit` - The element inherits the `float` value of its parent

In its simplest use, the `float` property can be used to wrap text around images.

### **Clear**

When we use the `float` property, and we want the next element below (not on right or left or both), we will have to use the `clear` property.

The clear property specifies what should happen with the element that is next to a floating element.



### The Clearfix Hack

```
.clearfix::after {  
  content: "";  
  clear: both;  
  display: table;  
}
```

## CSS Navigation Bar

### Def

A navigation bar is basically a list of links, so using the `<ul>` and `<li>` elements makes perfect sense:

```
<ul>  
  <li><a href="default.asp">Home</a></li>  
  <li><a href="news.asp">News</a></li>  
  <li><a href="contact.asp">Contact</a></li>  
  <li><a href="about.asp">About</a></li>  
</ul>  
  
ul {  
  list-style-type: none;  
  margin: 0;  
  padding: 0;  
}
```

- `list-style-type: none;` - Removes the bullets. A navigation bar does not need list markers
- Set `margin: 0;` and `padding: 0;` to remove browser default settings

### Vertical Navbar



```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 25%;
  background-color: #f1f1f1;
  height: 100%; /* Full height */
  position: fixed; /* Make it stick, even on scroll */
  overflow: auto; /* Enable scrolling if the sidenav has too much content */
}
```

## Horizontal Navbar



There are two ways to create a horizontal navigation bar. Using **inline** or **floating** list items.

### 1. Inline list Navbar

```
li {
  display: inline;
}
```

### 2. Floating list Navbar

```
li {
  float: left;
}

a {
  display: block;
  padding: 8px;
  background-color: #dddddd;
}
```

## CSS Dropdown Menu

**HTML**) Use any element to open the dropdown content, e.g. a `<span>`, or a `<button>` element.

Use a container element (like `<div>`) to create the dropdown content and add whatever you want inside of it.

Wrap a `<div>` element around the elements to position the dropdown content correctly with CSS.

**CSS**) The `.dropdown` class uses `position: relative`, which is needed when we want the dropdown content to be placed right below the dropdown button (using `position: absolute`).

The `.dropdown-content` class holds the actual dropdown content. It is hidden by default, and will be displayed on hover (see below). Note the `min-width` is set to 160px. Feel free to change this. **Tip:** If you want the width of the dropdown content to be as wide as the dropdown button, set the `width` to 100% (and `overflow: auto` to enable scroll on small screens).

Instead of using a border, we have used the CSS `box-shadow` property to make the dropdown menu look like a "card".

The `:hover` selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

# CSS Math Functions

## Def

| Function            | Description                                                                           |
|---------------------|---------------------------------------------------------------------------------------|
| <code>calc()</code> | Allows you to perform calculations to determine CSS property values                   |
| <code>max()</code>  | Uses the largest value, from a comma-separated list of values, as the property value  |
| <code>min()</code>  | Uses the smallest value, from a comma-separated list of values, as the property value |

### 1. Calc Function

Use `calc()` to calculate the width of a `<div>` element:

```
#div1 {  
  position: absolute;  
  left: 50px;  
  width: calc(100% - 100px);  
  border: 1px solid black;  
  background-color: yellow;  
  padding: 5px;  
}
```

### 2. Max Function

Use `max()` to set the width of `#div1` to whichever value is largest, 50% or 300px:

```
#div1 {  
  background-color: yellow;  
  height: 100px;  
  width: max(50%, 300px);  
}
```

### 3. Min Function

Use `min()` to set the width of `#div1` to whichever value is smallest, 50% or 300px:

```
#div1 {  
  background-color: yellow;  
  height: 100px;  
  width: min(50%, 300px);  
}
```



# CSS Units

## Def

CSS has several different units for expressing a length.

Many CSS properties take "length" values, such as width, margin, padding, font-size, etc.

**Length** is a number followed by a length unit, such as 10px, 2em, etc.

There are two types of length units: **absolute** and **relative**.

## Absolute lengths

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

| Unit | Description                  |                        |
|------|------------------------------|------------------------|
| cm   | centimeters                  | <a href="#">Try it</a> |
| mm   | millimeters                  | <a href="#">Try it</a> |
| in   | inches (1in = 96px = 2.54cm) | <a href="#">Try it</a> |
| px * | pixels (1px = 1/96th of 1in) | <a href="#">Try it</a> |
| pt   | points (1pt = 1/72 of 1in)   | <a href="#">Try it</a> |
| pc   | picas (1pc = 12 pt)          | <a href="#">Try it</a> |

\* Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

## Relative Lengths

Relative length units specify a length relative to another length property. Relative length units scale better between different rendering mediums.

| Unit | Description                                                                               |                        |
|------|-------------------------------------------------------------------------------------------|------------------------|
| em   | Relative to the font-size of the element (2em means 2 times the size of the current font) | <a href="#">Try it</a> |
| ex   | Relative to the x-height of the current font (rarely used)                                | <a href="#">Try it</a> |
| ch   | Relative to width of the "0" (zero)                                                       | <a href="#">Try it</a> |
| rem  | Relative to font-size of the root element                                                 | <a href="#">Try it</a> |
| vw   | Relative to 1% of the width of the viewport*                                              | <a href="#">Try it</a> |
| vh   | Relative to 1% of the height of the viewport*                                             | <a href="#">Try it</a> |
| vmin | Relative to 1% of viewport's* smaller dimension                                           | <a href="#">Try it</a> |
| vmax | Relative to 1% of viewport's* larger dimension                                            | <a href="#">Try it</a> |
| %    | Relative to the parent element                                                            | <a href="#">Try it</a> |

## CSS Transform – 2D

### Def

CSS transforms allow you to move, rotate, scale, and skew elements.

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

### 1. Translate() Method

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

```
div {  
  transform: translate(50px, 100px);  
}
```



### 2. Rotate() Method

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

```
div {  
  transform: rotate(20deg);  
}
```



### 3. Scale() Method

The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

```
div {  
  transform: scale(2, 3);  
}
```



- The scaleX() method increases or decreases the width of an element.
- The scaleY() method increases or decreases the height of an element.

#### 4. Skew() Method

The skew() method skews an element along the X and Y-axis by the given angles.

```
div {  
  transform: skew(20deg, 10deg);  
}
```

- The skewX() method skews an element along the X-axis by the given angle.
- The skewY() method skews an element along the Y-axis by the given angle.



#### 5. Matrix() Method

The matrix() method combines all the 2D transform methods into one.

The matrix() method takes six parameters, containing mathematical functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follows: matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())

```
div {  
  transform: matrix(1, -0.3, 0, 1, 0, 0);  
}
```



#### Summary

| Function              | Description                                                                |
|-----------------------|----------------------------------------------------------------------------|
| matrix(n,n,n,n,n,n)   | Defines a 2D transformation, using a matrix of six values                  |
| translate(x,y)        | Defines a 2D translation, moving the element along the X- and the Y-axis   |
| translateX(n)         | Defines a 2D translation, moving the element along the X-axis              |
| translateY(n)         | Defines a 2D translation, moving the element along the Y-axis              |
| scale(x,y)            | Defines a 2D scale transformation, changing the element's width and height |
| scaleX(n)             | Defines a 2D scale transformation, changing the element's width            |
| scaleY(n)             | Defines a 2D scale transformation, changing the element's height           |
| rotate(angle)         | Defines a 2D rotation, the angle is specified in the parameter             |
| skew(x-angle,y-angle) | Defines a 2D skew transformation along the X- and the Y-axis               |
| skewX(angle)          | Defines a 2D skew transformation along the X-axis                          |
| skewY(angle)          | Defines a 2D skew transformation along the Y-axis                          |

## CSS Transform – 3D

### Def

CSS also supports 3D transformations.

With the CSS transform property you can use the following 3D transformation methods:

- rotateX()
- rotateY()
- rotateZ()

### 1. rotateX() Method

The rotateX() method rotates an element around its X-axis at a given degree:

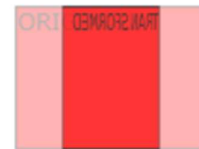
```
#myDiv {  
  transform: rotateX(150deg);  
}
```



### 2. rotateY() Method

The rotateY() method rotates an element around its Y-axis at a given degree:

```
# myDiv {  
  transform: rotateY(150deg);  
}
```



### 3. rotateZ() Method

The rotateZ() method rotates an element around its Z-axis at a given degree:

```
#myDiv {  
  transform: rotateZ(90deg);  
}
```

### Summary

| Property                            | Description                                                                    |
|-------------------------------------|--------------------------------------------------------------------------------|
| <a href="#">transform</a>           | Applies a 2D or 3D transformation to an element                                |
| <a href="#">transform-origin</a>    | Allows you to change the position on transformed elements                      |
| <a href="#">transform-style</a>     | Specifies how nested elements are rendered in 3D space                         |
| <a href="#">perspective</a>         | Specifies the perspective on how 3D elements are viewed                        |
| <a href="#">perspective-origin</a>  | Specifies the bottom position of 3D elements                                   |
| <a href="#">backface-visibility</a> | Defines whether or not an element should be visible when not facing the screen |

# CSS Transition

## Def

CSS transitions allows you to change property values smoothly, over a given duration.

It consist of following properties:

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

## Summary

| Property                                          | Description                                                                            |
|---------------------------------------------------|----------------------------------------------------------------------------------------|
| <a href="#"><u>transition</u></a>                 | A shorthand property for setting the four transition properties into a single property |
| <a href="#"><u>transition-delay</u></a>           | Specifies a delay (in seconds) for the transition effect                               |
| <a href="#"><u>transition-duration</u></a>        | Specifies how many seconds or milliseconds a transition effect takes to complete       |
| <a href="#"><u>transition-property</u></a>        | Specifies the name of the CSS property the transition effect is for                    |
| <a href="#"><u>transition-timing-function</u></a> | Specifies the speed curve of the transition effect                                     |

## Transition Shorthand Property

The CSS transition properties can be specified one by one, like this:

```
div {  
  transition-property: width;  
  transition-duration: 2s;  
  transition-timing-function: linear;  
  transition-delay: 1s;  
}
```

or by using the shorthand property transition:

```
div {  
  transition: width 2s linear 1s;  
}
```

## Transition Timing Function

The transition-timing-function property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- linear - specifies a transition effect with the same speed from start to end
- ease-in - specifies a transition effect with a slow start
- ease-out - specifies a transition effect with a slow end
- ease-in-out - specifies a transition effect with a slow start and end
- cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

### **Transition + Transformation**

```
div {  
  
    width: 100px;  
  
    height: 100px;  
  
    background: red;  
  
    transition: width 2s, height 2s, transform 2s;  
  
}  
  
div:hover {  
  
    width: 300px;  
  
    height: 300px;  
  
    transform: rotate(180deg);  
  
}
```

# CSS Animation

## Def

CSS allows animation of HTML elements without using JavaScript!

It consists of the following properties:

- @keyframes
- animation-name
- animation-duration
- animation-delay
- animation-iteration-count
- animation-direction
- animation-timing-function
- animation-fill-mode
- animation

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

## Summary

| Property                                  | Description                                                                                                    |
|-------------------------------------------|----------------------------------------------------------------------------------------------------------------|
| <a href="#">@keyframes</a>                | Specifies the animation code                                                                                   |
| <a href="#">animation</a>                 | A shorthand property for setting all the animation properties                                                  |
| <a href="#">animation-delay</a>           | Specifies a delay for the start of an animation                                                                |
| <a href="#">animation-direction</a>       | Specifies whether an animation should be played forwards, backwards or in alternate cycles                     |
| <a href="#">animation-duration</a>        | Specifies how long time an animation should take to complete one cycle                                         |
| <a href="#">animation-fill-mode</a>       | Specifies a style for the element when the animation is not playing (before it starts, after it ends, or both) |
| <a href="#">animation-iteration-count</a> | Specifies the number of times an animation should be played                                                    |
| <a href="#">animation-name</a>            | Specifies the name of the @keyframes animation                                                                 |
| <a href="#">animation-play-state</a>      | Specifies whether the animation is running or paused                                                           |
| <a href="#">animation-timing-function</a> | Specifies the speed curve of the animation                                                                     |

## Animation shorthand Property

The example below uses six of the animation properties:

```
div {  
  animation-name: example;  
  animation-duration: 5s;  
  animation-timing-function: linear;  
  animation-delay: 2s;  
  animation-iteration-count: infinite;  
  animation-direction: alternate;  
}
```

The same animation effect as above can be achieved by using the shorthand animation property:

```
div {  
  animation: example 5s linear 2s infinite alternate;  
}
```

### **@keyframe Rule**

When you specify CSS styles inside the @keyframes rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

```
/* The animation code */  
@keyframes example {  
  from {background-color: red;}  
  to {background-color: yellow;}  
}  
  
/* The element to apply the animation to */  
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
}
```

It is also possible to use percent. By using percent, you can add as many style changes as you like.

```
/* The animation code */  
@keyframes example {  
  0% {background-color: red;}  
  25% {background-color: yellow;}  
  50% {background-color: blue;}  
  100% {background-color: green;}  
}
```