CSS Tutorial

CSS is the language we use to style an HTML document.

CSS describes how HTML elements should be displayed.

## The CSS Universal Selector

The universal selector (\*) selects all HTML elements on the page.

\* {

    text-align: center;

    color: blue;

  }

## The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

h1,

h2,

p {

    text-align: center;

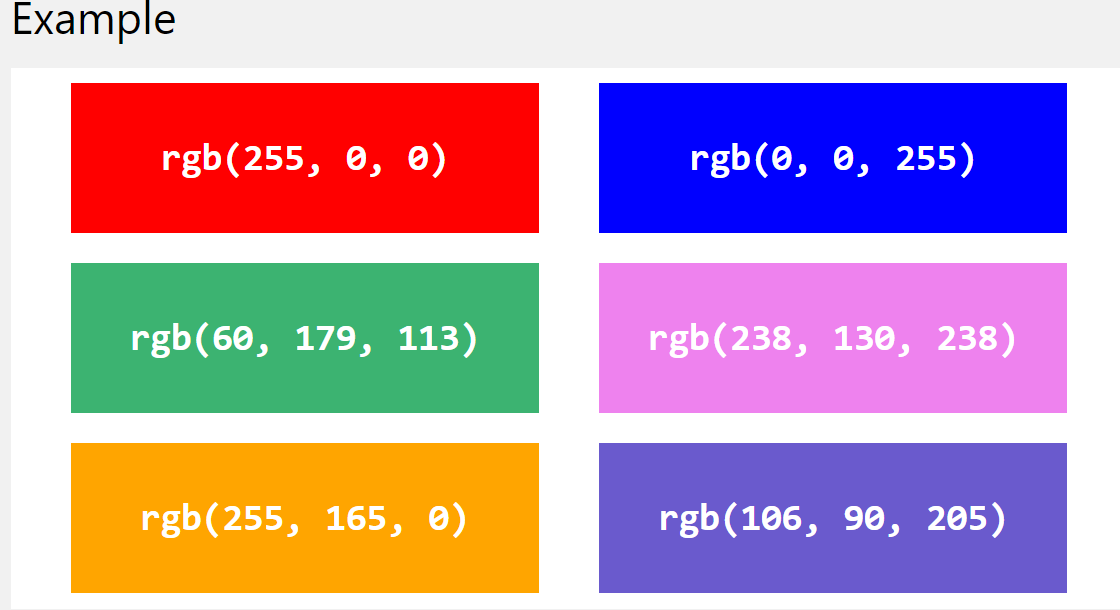
    color: red;

}

## RGB Value

In CSS, a color can be specified as an RGB value, using this formula:

**rgb(red, green, blue)**

****

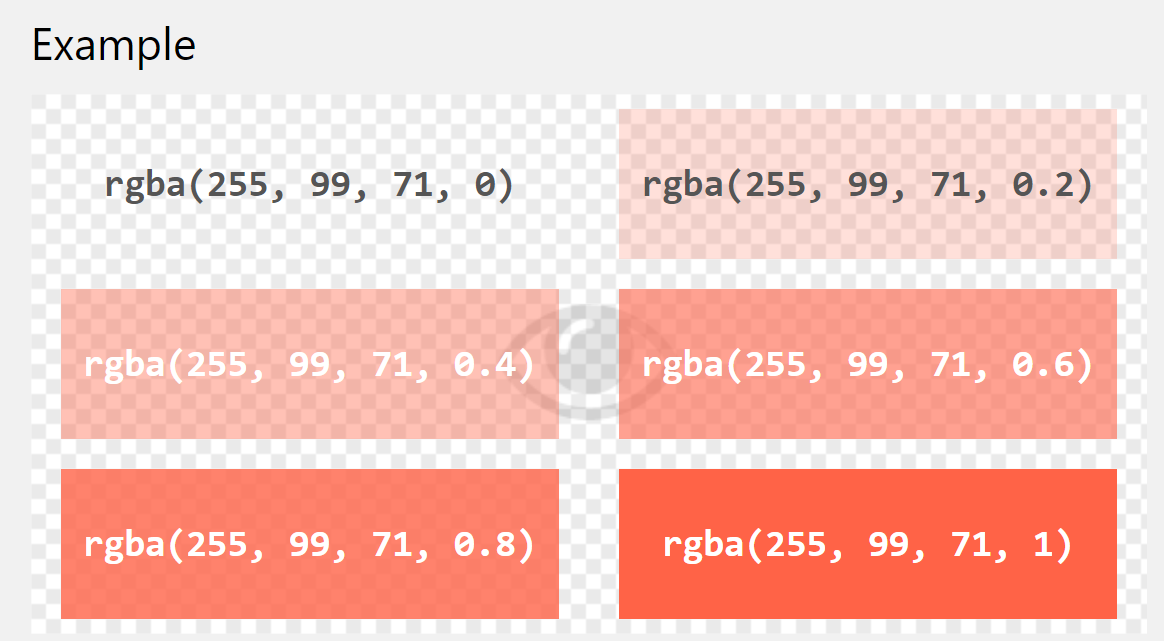
## RGBA Value

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.

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 (not transparent at all):



CSS HEX Colors

[❮ Previous](https://www.w3schools.com/css/css_colors_rgb.asp)[Next ❯](https://www.w3schools.com/css/css_colors_hsl.asp)

A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.

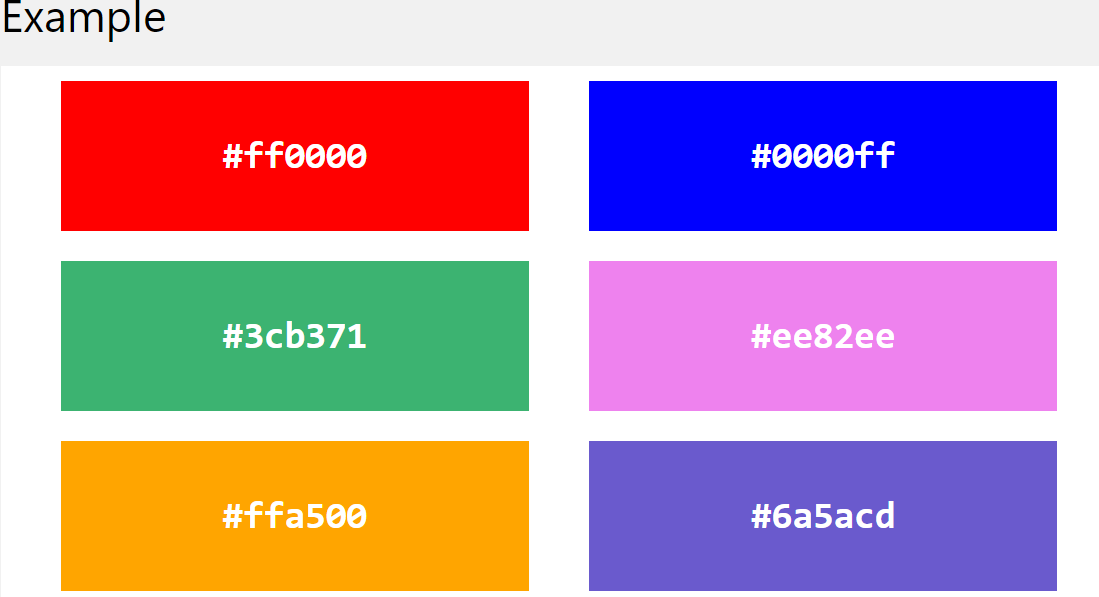
## HEX Value

In CSS, a color can be specified using a hexadecimal value in the form:

**#rrggbb**

**#ff6347**

## 3 Digit HEX Value



Sometimes you will see a 3-digit hex code in the CSS source.

The 3-digit hex code is a shorthand for some 6-digit hex codes.

The 3-digit hex code has the following form:

**#rgb**

Where r, g, and b represents the red, green, and blue components with values between 0 and f.

CSS HSL Colors

[❮ Previous](https://www.w3schools.com/css/css_colors_hex.asp)[Next ❯](https://www.w3schools.com/css/css_background.asp)

HSL stands for hue, saturation, and lightness.

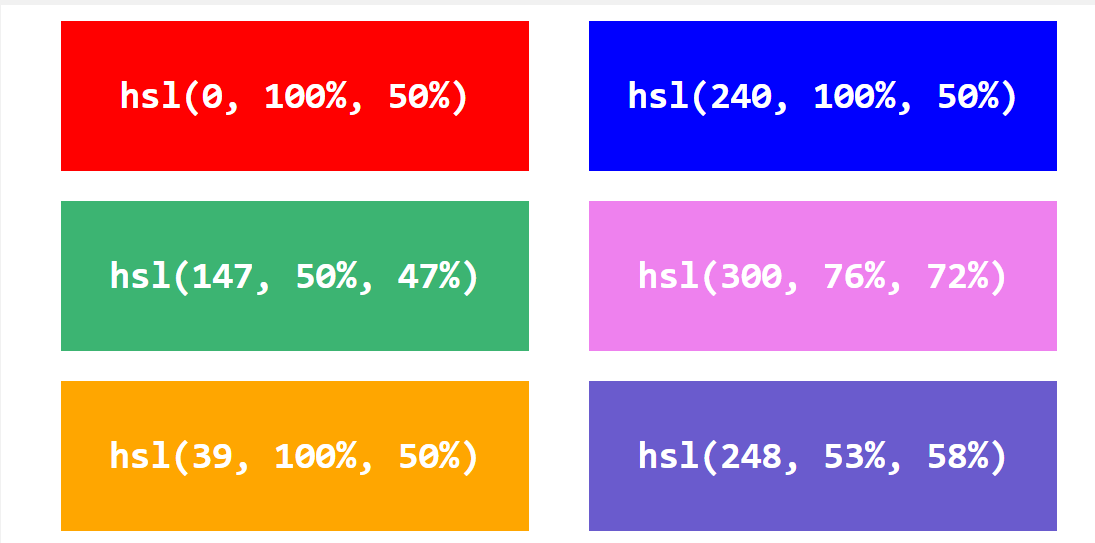
**hsl(hue, saturation, lightness)**

Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.

Saturation is a percentage value, 0% means a shade of gray, and 100% is the full color.

Lightness is also a percentage, 0% is black, 50% is neither light or dark, 100% is white

**hsl(0, 100%, 50%)**

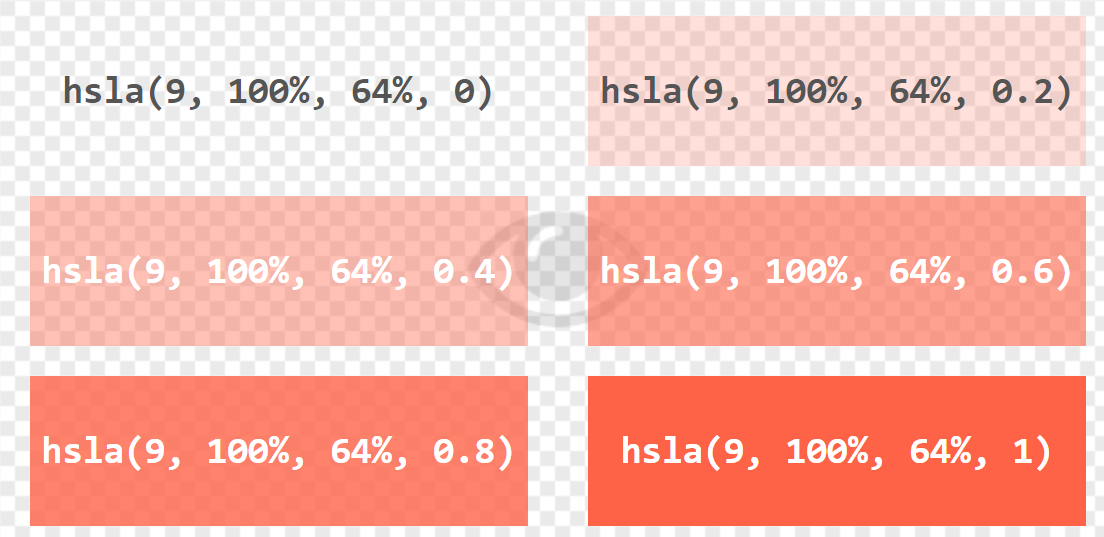


## HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

**hsla(hue, saturation, lightness, alpha)**



## CSS background-attachment

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):

body {

    background-image: url("img\_tree.png");

    background-repeat: no-repeat;

    background-position: right top;

    background-attachment: fixed;

  }

*/\* with scroll \*/*

  body {

    background-image: url("img\_tree.png");

    background-repeat: no-repeat;

    background-position: right top;

    background-attachment: scroll;

  }

## CSS background - Shorthand property

body {

    background: #ffffff url("img\_tree.png") no-repeat right top;

}

CSS Borders

[❮ Previous](https://www.w3schools.com/css/css_background_shorthand.asp)[Next ❯](https://www.w3schools.com/css/css_border_width.asp)

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

## CSS 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. The effect depends on the border-color value
* ridge - Defines a 3D ridged border. The effect depends on the border-color value
* inset - Defines a 3D inset border. The effect depends on the border-color value
* outset - Defines a 3D outset border. The effect depends on the border-color value
* none - Defines no border
* hidden - Defines a hidden border

## CSS Border Width

p.one {

    border-style: solid;

    border-width: 5px;

}

p.two {

    border-style: solid;

    border-width: medium;

}

p.three {

    border-style: dotted;

    border-width: 2px;

}

p.four {

    border-style: dotted;

    border-width: thick;

}

p.five {

    border-style: double;

    border-width: 15px;

}

p.six {

    border-style: double;

    border-width: thick;

}

## CSS Border - Shorthand Property

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;

}

p {

    border-left: 6px solid red;

    background-color: lightgrey;

}

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

p {

    border: 2px solid red;

    border-radius: 5px;}

CSS Margins

Margins 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

**Tip:** Negative values are allowed.

## Margin - Shorthand Property

**margin: 25px 50px 75px 100px;**

* top margin is 25px
* right margin is 50px
* bottom margin is 75px
* left margin is 100px

p {

    margin: 25px 50px 75px 100px;

}

If the margin property has three values:

* **margin: 25px 50px 75px;**
  + top margin is 25px
  + **right and left margins are 50px**
  + bottom margin is 75px

If the margin property has two values:

* **margin: 25px 50px;**
  + top and bottom margins are 25px
  + right and left margins are 50px

If the margin property has one value:

* **margin: 25px;**
  + all four margins are 25px

## The auto Value

You can set the margin property to auto to horizontally center the element within its container.

div {

    width: 300px;

    margin: auto;

    border: 1px solid red;

}

## The inherit Value

This example lets the left margin of the <p class="ex1"> element be inherited from the parent element (<div>):

div {

    border: 1px solid red;

    margin-left: 100px;

}

p.ex1 {

    margin-left: inherit;

}

# CSS Margin Collapse

Sometimes two margins collapse into a single margin.

h1 {

    margin: 0 0 50px 0;

}

h2 {

    margin: 20px 0 0 0;

}

# CSS Padding

Padding is used to create space around an element's content, inside of any defined borders.

div {

    padding: 70px;

    border: 1px solid #4CAF50;

}

## Padding - Individual Sides

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

div {

    border: 1px solid black;

    background-color: lightblue;

    padding-top: 50px;

    padding-right: 30px;

    padding-bottom: 50px;

    padding-left: 80px;

}

## Padding - Shorthand Property

**padding: 25px 50px 75px 100px;**

div {

    padding: 25px 50px 75px 100px;

}

If the padding property has three values:

* **padding: 25px 50px 75px;**
  + top padding is 25px
  + right and left paddings are 50px
  + bottom padding is 75px

div {

    padding: 25px 50px 75px;

}

If the padding property has two values:

* **padding: 25px 50px;**
  + top and bottom paddings are 25px
  + right and left paddings are 50px

div {

    padding: 25px 50px; }

If the padding property has one value:

* **padding: 25px;**
  + all four paddings are 25px

## Padding and Element Width

Here, the <div> element is given a width of 300px. However, the actual width of the <div> element will be 350px (300px + 25px of left padding + 25px of right padding):

To keep the width at 300px, no matter the amount of padding, you can use the box-sizing property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease.

div {

    width: 300px;

    padding: 25px;

    box-sizing: border-box;

}

# CSS Height and 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.

## CSS height and width Values

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

## Setting max-width

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

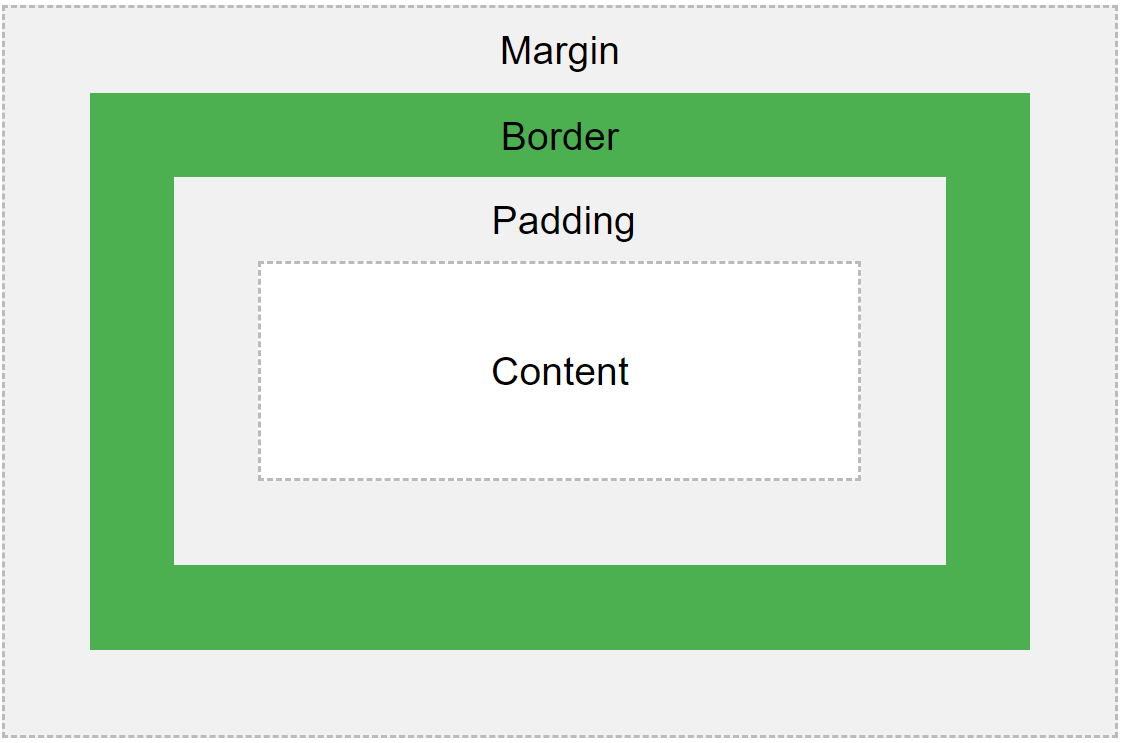
# CSS Box Model

All HTML elements can be considered as boxes.

## The CSS Box Model

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: margins, borders, padding, and the actual content. The image below illustrates the box model:



* **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

div {

    background-color: lightgrey;

    width: 300px;

    border: 15px solid green;

    padding: 50px;

    margin: 20px;

}

## Width and Height of an Element

**Important:** When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the full size of an element, you must also add padding, borders and margins.

div {

    width: 320px;

    padding: 10px;

    border: 5px solid gray;

    margin: 0;

}

320px (width)  
+ 20px (left + right padding)  
+ 10px (left + right border)  
+ 0px (left + right margin)  
**= 350px**

# CSS Outline

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

p {

    border: 2px solid black;

    outline: #4CAF50 solid 10px;

    margin: auto;

    padding: 20px;

    text-align: center;

}

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".

CSS has the following outline properties:

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

**Note:** Outline differs from [borders](https://www.w3schools.com/css/css_border.asp)! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

## CSS Outline Style

The outline-style property specifies the style of the outline, and can have one of the following values:

* dotted - Defines a dotted outline
* dashed - Defines a dashed outline
* solid - Defines a solid outline
* double - Defines a double outline
* groove - Defines a 3D grooved outline
* ridge - Defines a 3D ridged outline
* inset - Defines a 3D inset outline
* outset - Defines a 3D outset outline
* none - Defines no outline
* hidden - Defines a hidden outline

p {

    outline-color: red;

}

p.dotted {

    outline-style: dotted;

}

p.dashed {

    outline-style: dashed;

}

p.solid {

    outline-style: solid;

}

p.double {

    outline-style: double;

}

p.groove {

    outline-style: groove;

}

p.ridge {

    outline-style: ridge;

}

p.inset {

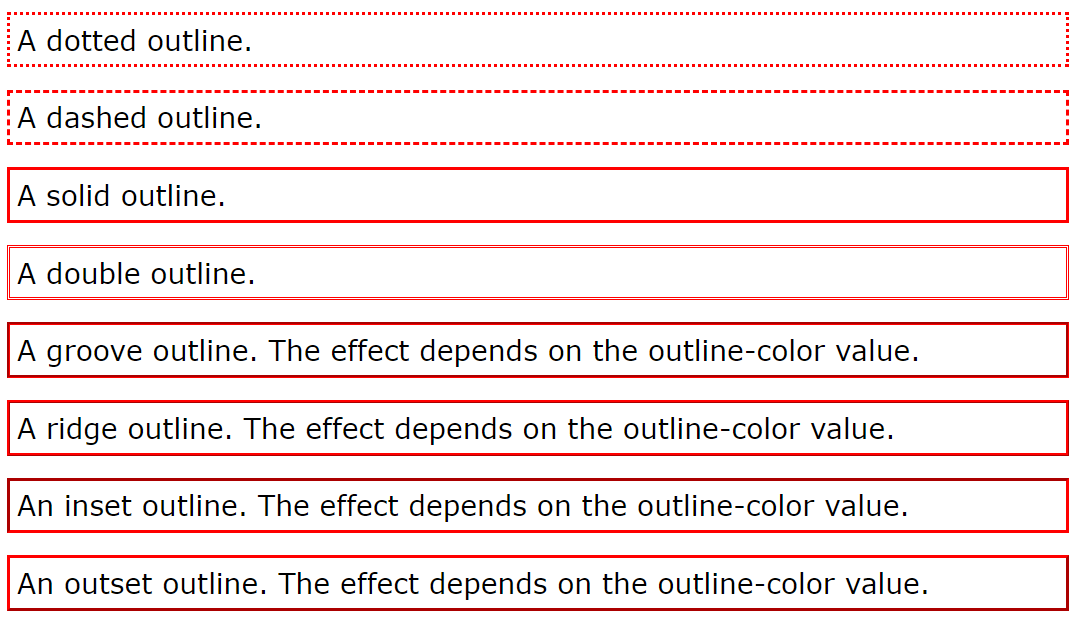
    outline-style: inset;

}

p.outset {

    outline-style: outset;

}



## CSS Outline Width

The outline-width property specifies the width of the outline, and can have one of the following values:

* thin (typically 1px)
* medium (typically 3px)
* thick (typically 5px)
* A specific size (in px, pt, cm, em, etc)

p.ex1 {

    border: 1px solid black;

    outline-style: solid;

    outline-color: red;

    outline-width: thin;

}

p.ex2 {

    border: 1px solid black;

    outline-style: solid;

    outline-color: red;

    outline-width: medium;

}

p.ex3 {

    border: 1px solid black;

    outline-style: solid;

    outline-color: red;

    outline-width: thick;

}

p.ex4 {

    border: 1px solid black;

    outline-style: solid;

    outline-color: red;

    outline-width: 4px;

}



## CSS Outline Color

The outline-color property is used to set the color of the outline.

* 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%)"
* invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

## Invert Color

The following example uses outline-color: invert, which performs a color inversion. This ensures that the outline is visible, regardless of color background:

p.ex1 {

    outline-style: solid;

    outline-color: #ff0000;

*/\* red \*/*

}

p.ex2 {

    outline-style: solid;

    outline-color: rgb(255, 0, 0);

*/\* red \*/*

}

p.ex3 {

    outline-style: solid;

    outline-color: hsl(0, 100%, 50%);

*/\* red \*/*

}

p.ex4 {

    border: 1px solid yellow;

    outline-style: solid;

    outline-color: invert;

}

## CSS Outline - Shorthand property

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

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

p.ex1 {outline: dashed;}

p.ex2 {outline: dotted red;}

p.ex3 {outline: 5px solid yellow;}

p.ex4 {outline: thick ridge pink;}

## CSS 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.

p {

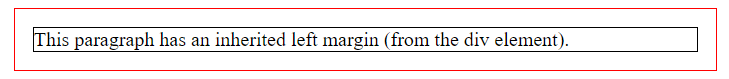
    margin: 30px;

    border: 1px solid black;

    outline: 1px solid red;

    outline-offset: 15px;

}



CSS Text

CSS has a lot of properties for formatting text.

<div>

        <h1>text formatting</h1>

        <p>This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed

            from this colored

            <a *target*="\_blank" *href*="tryit.asp?filename=trycss\_text">"Try it Yourself"</a> link.

        </p>

    </div>

CSS

div {

    border: 1px solid gray;

    padding: 8px;

}

h1 {

    text-align: center;

    text-transform: uppercase;

    color: #4CAF50;

}

p {

    text-indent: 50px;

    text-align: justify;

    letter-spacing: 3px;

}

a {

    text-decoration: none;

    color: #008CBA;

}

## Text Alignment

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

h1 {

    text-align: center;

}

h2 {

    text-align: left;

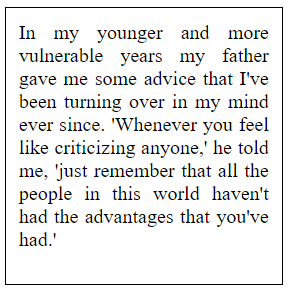
}

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 straight (like in magazines and newspapers):



## Text Direction

The direction and unicode-bidi properties can be used to change the text direction of an element:

p.ex1 {

    direction: rtl;

    unicode-bidi: bidi-override;

}

This is the default text direction.

This is the default text direction.

## Vertical Alignment

The vertical-align property sets the vertical alignment of an element.

## 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:

a {

    text-decoration: none;

}

h1 {

    text-decoration: overline;

}

h2 {

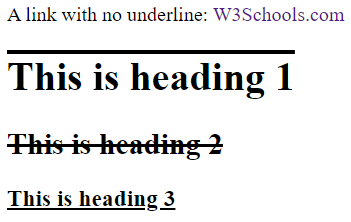
    text-decoration: line-through;

}

h3 {

    text-decoration: underline;

}



## Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

p.uppercase {

    text-transform: uppercase;

}

p.lowercase {

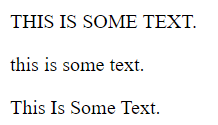
    text-transform: lowercase;

}

p.capitalize {

    text-transform: capitalize;

}



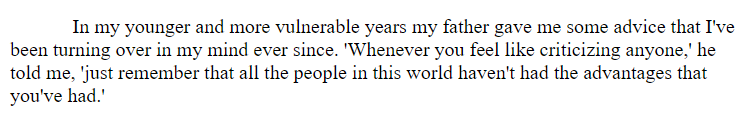
## Text Indentation

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

p {

    text-indent: 50px;

}



## Letter Spacing

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

h1 {

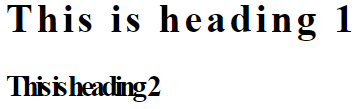
    letter-spacing: 3px;

}

h2 {

    letter-spacing: -3px;

}



## Line Height

The line-height property is used to specify the space between lines:

p.small {

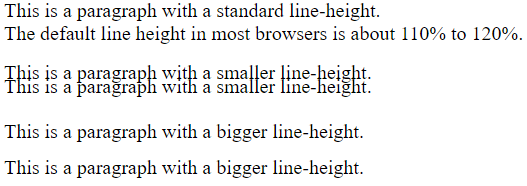
    line-height: 0.7;

}

p.big {

    line-height: 1.8;

}



## Word Spacing

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

h1 {

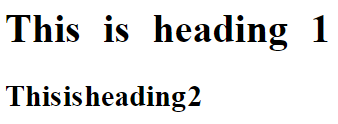
    word-spacing: 10px;

}

h2 {

    word-spacing: -5px;

}



## White Space

The white-space property specifies how white-space inside an element is handled.

p {

    white-space: nowrap;

}

## Text Shadow

The text-shadow property adds shadow to text.

*/\*horizontal shadow (2px) and the vertical shadow (2px):\*/*

.shadow {

    text-shadow: 2px 2px;

}

*/\* add a color (red) to the shadow:\*/*

.shadow-color {

    text-shadow: 2px 2px red;

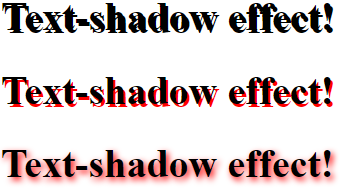
}

*/\* add a blur effect (5px) to the shadow: \*/*

.shadow-blur {

    text-shadow: 2px 2px 5px red;

}



# CSS Fonts

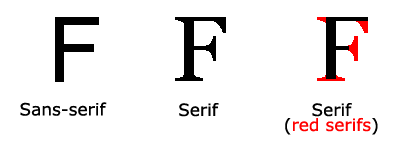
Choosing the right font for your website is important!

## 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.

## Difference Between Serif and Sans-serif Fonts



**Note:** On computer screens, sans-serif fonts are considered easier to read than serif fonts.

## The CSS font-family Property

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

If the font name is more than one word, it must be in quotation marks, like: "Times New Roman".

.p1 {

    font-family: "Times New Roman", Times, serif;

}

.p2 {

    font-family: Arial, Helvetica, sans-serif;

}

.p3 {

    font-family: "Lucida Console", "Courier New", monospace;

}

This is a paragraph, shown in the Times New Roman font.

This is a paragraph, shown in the Arial font.

This is a paragraph, shown in the Lucida Console font.

CSS Web Safe Fonts[❮ Previous](https://www.w3schools.com/css/css_font.asp)[Next ❯](https://www.w3schools.com/css/css_font_fallbacks.asp)

## What are Web Safe Fonts?

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

## Best Web Safe Fonts for HTML and CSS

The following list are the best web safe fonts for HTML and CSS:

* Arial (sans-serif)
* Verdana (sans-serif)
* Helvetica (sans-serif)
* Tahoma (sans-serif)
* Trebuchet MS (sans-serif)
* Times New Roman (serif)
* Georgia (serif)
* Garamond (serif)
* Courier New (monospace)
* Brush Script MT (cursive)

**Note:** Before you publish your website, always check how your fonts appear on different browsers and devices, and always use fallback fonts!

# CSS Font Fallbacks

## Commonly Used Font Fallbacks

Below are some commonly used font fallbacks, organized by the 5 generic font families:

* **Serif**
* **Sans-serif**
* **Monospace**
* **Cursive**
* **Fantasy**

## 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)

p.normal {

    font-style: normal;

}

p.italic {

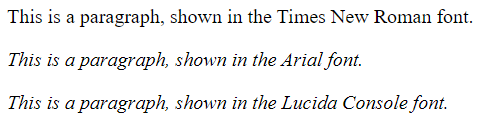
    font-style: italic;

}

p.oblique {

    font-style: oblique;

}



## Font Weight

The font-weight property specifies the weight of a font:

p.normal {

    font-weight: normal;

}

p.light {

    font-weight: lighter;

}

p.thick {

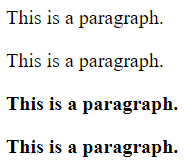
    font-weight: bold;

}

p.thicker {

    font-weight: 900;

}



## Font Variant

The font-variant property specifies whether or not a text should be displayed in a small-caps font.

p.normal {

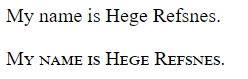
    font-variant: normal;

}

p.small {

    font-variant: small-caps;

}



## Font Size

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

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 (16px=1em).

## Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

h1 {

    font-size: 40px;

}

h2 {

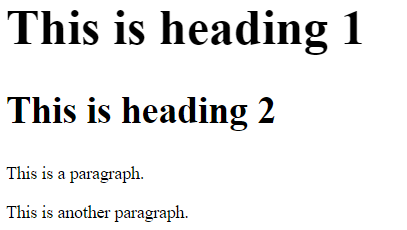
    font-size: 30px;

}

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.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

h1 {

    font-size: 2.5em;

*/\* 40px/16=2.5em \*/*

}

h2 {

    font-size: 1.875em;

*/\* 30px/16=1.875em \*/*

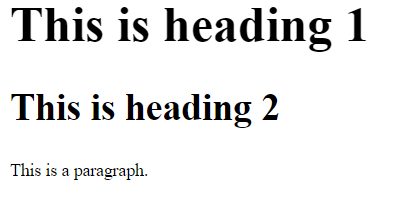
}

p {

    font-size: 0.875em;

*/\* 14px/16=0.875em \*/*

}



## Responsive Font Size

The text size can be set with a vw unit, which means the "viewport width".

<h1 style="**font-size:10vw**">Hello World</h1>

<meta *name*="viewport" *content*="width=device-width, initial-scale=1.0">

<body>

    <h1 *style*="font-size:10vw;">Responsive Text</h1>

    <p *style*="font-size:5vw;">Resize the browser window to see how the text size scales.</p>

    <p *style*="font-size:5vw;">Use the "vw" unit when sizing the text. 10vw will set the size to 10% of the viewport width.

    </p>

</body>

Viewport is the browser window size. 1vw = 1% of viewport width. If the viewport is 50cm wide, 1vw is 0.5cm.

## Google Fonts

If you do not want to use any of the standard fonts in HTML, you can use Google Fonts.

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

## How To Use Google Fonts

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">

<style>

    body {

        font-family: "Sofia", sans-serif;

    }

</style>

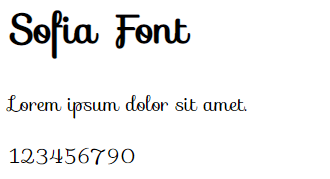
</head>

<body>

    <h1>Sofia Font</h1>

    <p>Lorem ipsum dolor sit amet.</p>

    <p>123456790</p>



## Use Multiple Google Fonts

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

<link *rel*="stylesheet" *href*="https://fonts.googleapis.com/css?family=Audiowide|Sofia|Trirong">

<style>

    h1.a {

        font-family: "Audiowide", sans-serif;

    }

    h1.b {

        font-family: "Sofia", sans-serif;

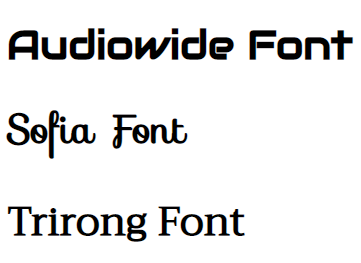
    }

    h1.c {

        font-family: "Trirong", serif;

    }

</style>



## Styling Google Fonts

Of course you can style Google Fonts as you like, with CSS!

<style>

    body {

        font-family: "Sofia", sans-serif;

        font-size: 30px;

        text-shadow: 3px 3px 3px #ababab;

    }

</style>



## Enabling Font Effects

Google have also enabled different font effects that you can use.

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.

<link *rel*="stylesheet" *href*="https://fonts.googleapis.com/css?family=Sofia&effect=fire">

<style>

    body {

        font-family: "Sofia", sans-serif;

        font-size: 30px;

    }

</style>

</head>

<body>

    <h1 *class*="**font-effect-fire**">Sofia on Fire</h1>

</body>



To request multiple font effects, just separate the effect names with a pipe character (|), like this:

<head>

    <link *rel*="stylesheet" *href*="https://fonts.googleapis.com/css?family=Sofia&effect=neon|outline|emboss|shadow-multiple">

    <style>

        body {

            font-family: "Sofia", sans-serif;

            font-size: 30px;

        }

    </style>

</head>

<body>

    <h1 *class*="font-effect-neon">Neon Effect</h1>

    <h1 *class*="font-effect-outline">Outline Effect</h1>

    <h1 *class*="font-effect-emboss">Emboss Effect</h1>

    <h1 *class*="font-effect-shadow-multiple">Multiple Shadow Effect</h1>

</body>



CSS Great Font Pairings

Great font pairings are essential to great design.

## The CSS Font Property

To shorten the code, it is also possible to specify all the individual font properties in one 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 are used.

p.a {

    font: 20px Arial, sans-serif;

}

p.b {

    font: italic bold 12px/30px Georgia, serif;

}

# CSS Icons

Icons can easily be added to your HTML page, by using an icon library.

## How To Add Icons

The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

## Font Awesome Icons

<script src="https://kit.fontawesome.com/yourcode.js" crossorigin="anonymous"></script>

<!DOCTYPE *html*>

<html>

<head>

    <script *src*="https://kit.fontawesome.com/a076d05399.js" *crossorigin*="anonymous"></script>

</head>

<body>

    <i *class*="fas fa-cloud"></i>

    <i *class*="fas fa-heart"></i>

    <i *class*="fas fa-car"></i>

    <i *class*="fas fa-file"></i>

    <i *class*="fas fa-bars"></i>

</body>

</html>

## Bootstrap Icons

To use the Bootstrap glyphicons, add the following line inside the <head> section of your HTML page:

<!DOCTYPE *html*>

<html>

<head>

    <link *rel*="stylesheet" *href*="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

</head>

<body>

    <i *class*="glyphicon glyphicon-cloud"></i>

    <i *class*="glyphicon glyphicon-remove"></i>

    <i *class*="glyphicon glyphicon-user"></i>

    <i *class*="glyphicon glyphicon-envelope"></i>

    <i *class*="glyphicon glyphicon-thumbs-up"></i>

</body>

</html>

## Google Icons

To use the Google icons, add the following line inside the <head> section of your HTML page:

<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">

<!DOCTYPE *html*>

<html>

<head>

    <link *rel*="stylesheet" *href*="https://fonts.googleapis.com/icon?family=Material+Icons">

</head>

<body>

    <i *class*="material-icons">cloud</i>

    <i *class*="material-icons">favorite</i>

    <i *class*="material-icons">attachment</i>

    <i *class*="material-icons">computer</i>

    <i *class*="material-icons">traffic</i>

</body>

</html>

CSS Links

With CSS, links can be styled in many different ways.

## Styling Links

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

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

a:link {

    color: red;

}

*/\* visited link \*/*

a:visited {

    color: green;

}

*/\* mouse over link \*/*

a:hover {

    color: hotpink;

}

*/\* selected link \*/*

a:active {

    color: blue;

}

When setting the style for several link states, there are some order rules:

* a:hover MUST come after a:link and a:visited
* a:active MUST come after a:hover

a:link {

    text-decoration: none;

}

a:visited {

    text-decoration: none;

}

a:hover {

    text-decoration: underline;

}

a:active {

    text-decoration: underline;

}

## Background Color

a:link {

    background-color: yellow;

  }

  a:visited {

    background-color: cyan;

  }

  a:hover {

    background-color: lightgreen;

  }

  a:active {

    background-color: hotpink;

  }

## Link Buttons

This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

a:link,

a:visited {

    background-color: #f44336;

    color: white;

    padding: 14px 25px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

}

a:hover,

a:active {

    background-color: red;

}

CSS Layout - The display Property

The display property is the most important CSS property for controlling layout.

The display property specifies if/how an element is displayed.

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.

## 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 Elements

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;

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them.

## Override The Default Display Value

A common example is making inline <li> elements for horizontal menus

li {

    display: inline;

}



**Note:** Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with display: block; is not allowed to have other block elements inside it.

## Hide an Element - display:none or 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:

  h1.hidden {

            display: none;

        }

        h2 {

            visibility: hidden;

        }

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

div.ex1 {

    width: 500px;

    margin: auto;

    border: 3px solid #73AD21;

}

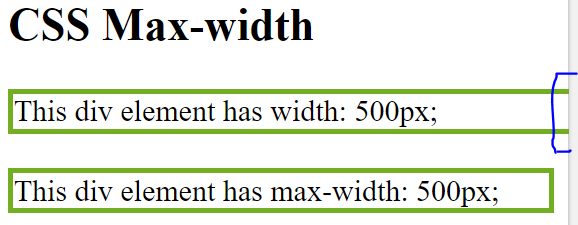
div.ex2 {

    max-width: 500px;

    margin: auto;

    border: 3px solid #73AD21;

}



CSS Layout - The position Property

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

There are five different position values:

* static
* relative
* fixed
* absolute
* 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.

## position: static;

HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

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: ;

    border: 3px solid #73AD21;

}

## position: relative;

An element with position: relative; is positioned relative to its normal position.

div.relative {

    position: relative;

    left: 30px;

    border: 3px solid #73AD21;

}

Other content will not be adjusted to fit into any gap left by the element.

## position: fixed;

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.

div.fixed {

    position: fixed;

    bottom: 0;

    right: 0;

    width: 300px;

    border: 3px solid #73AD21;

}

## position: absolute;

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

**Note:** A "positioned" element is one whose position is anything except static.

div.relative {

    position: relative;

    width: 400px;

    height: 200px;

    border: 3px solid #73AD21;

}

div.absolute {

    position: absolute;

    top: 60px;

    right: 0;

    width: 200px;

    height: 100px;

    border: 3px solid #73AD21;

}

## position: sticky;

An element with position: sticky; is positioned based on the user's scroll 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;

    background-color: green;

    border: 2px solid #4CAF50;

  }

## Overlapping Elements

When elements are positioned, they can overlap other elements.

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

img {

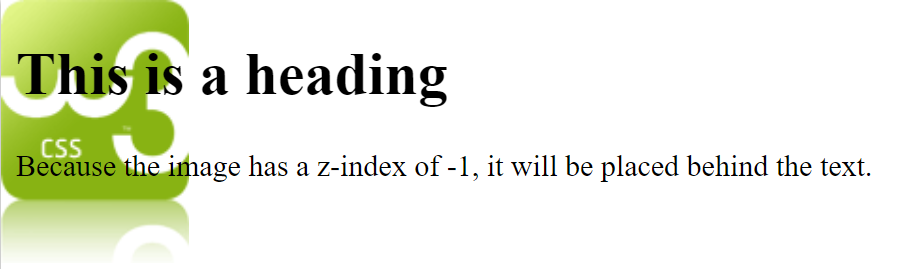
    position: absolute;

    left: 0px;

    top: 0px;

    z-index: -1;

}



## Positioning Text In an Image

How to position text over an image:

.container {

    position: relative;

}

.center {

    position: absolute;

    top: 50%;

    width: 100%;

    text-align: center;

    font-size: 18px;

}

img {

    width: 100%;

    height: auto;

    opacity: .3;

}

# CSS Layout - Overflow

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

#overflowTest {

    background: #4CAF50;

    color: white;

    padding: 15px;

    width: 50%;

    height: 100px;

    overflow: scroll;

    border: 1px solid #ccc;

}

## CSS Overflow

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.

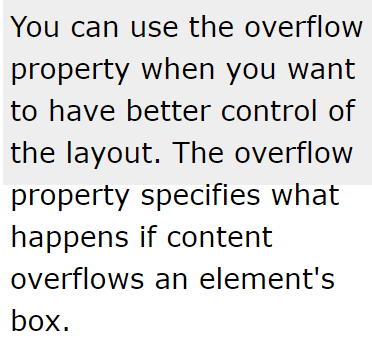
The overflow property has the following values:

* visible - Default. The overflow is not clipped. 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

**Note:** The overflow property only works for block elements with a specified height.

## overflow: visible

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



div {

    width: 200px;

    height: 50px;

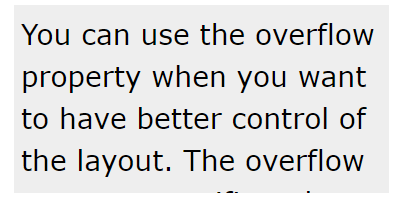
    background-color: #eee;

    overflow: visible;

}

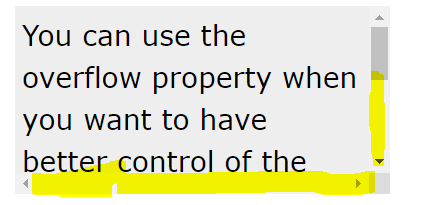
## overflow: hidden

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



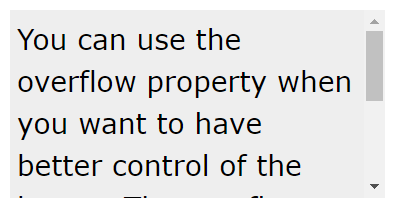
## overflow: scroll

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):



## overflow: auto

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



## overflow-x and overflow-y

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

# CSS Layout - float and clear

The CSS float property specifies how an element should float.

The CSS clear property specifies what elements can float beside the cleared element and on which side.

## The float Property

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.

# CSS Layout - clear and clearfix

## The clear Property

The clear property specifies what elements can float beside the cleared element and on which side.

The clear property can have one of the following values:

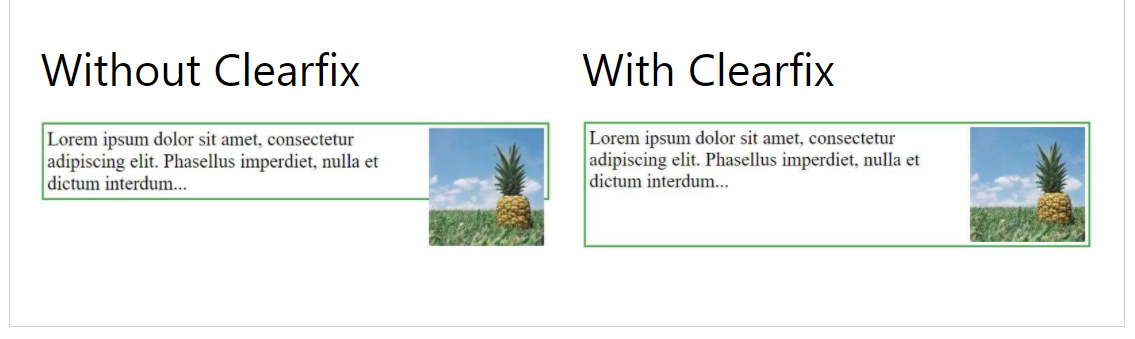
* none - Allows floating elements on both sides. This is default
* left - No floating elements allowed on the left side
* right- No floating elements allowed on the right side
* both - No floating elements allowed on either the left or the right side
* inherit - The element inherits the clear value of its parent

The most common way to use the clear property is after you have used a float property on an element.

When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

## The clearfix Hack

If an element is taller than the element containing it, and it is floated, it will "overflow" outside of its container:



Then we can add overflow: auto; to the containing element to fix this problem:

The overflow: auto clearfix works well as long as you are able to keep control of your margins and padding (else you might see scrollbars). The **new, modern clearfix hack** however, is safer to use, and the following code is used for most webpages:

div {

    border: 3px solid #4CAF50;

    padding: 5px;

}

.img1 {

    float: right;

}

.clearfix::after {

    content: "";

    clear: both;

    display: table;

}

.img2 {

    float: right;

}

# CSS Layout - Float Examples

# With the float property, it is easy to float boxes of content side by side:

\* {

    box-sizing: border-box;

}

.box {

    float: left;

    width: 33.33%;

    padding: 50px;

}

.clearfix::after {

    content: "";

    clear: both;

    display: table;

}

**What is box-sizing?**

You can easily create three floating boxes side by side. However, when you add something that enlarges the width of each box (e.g. padding or borders), the box will break. The box-sizing property allows us to include the padding and border in the box's total width (and height), making sure that the padding stays inside of the box and that it does not break.

## Equal Height Boxes

 it is not easy to create floating boxes with equal heights.

**However**, this is not very flexible. It is ok if you can guarantee that the boxes will always have the same amount of content in them. But many times, the content is not the same. If you try the example above on a mobile phone, you will see that the second box's content will be displayed outside of the box. This is where CSS3 Flexbox comes in handy - as it can automatically stretch boxes to be as long as the longest box:

Using **Flexbox** to create flexible boxes:

.flex-container {

    display: flex;

    flex-wrap: nowrap;

    background-color: DodgerBlue;

}

.flex-container .box {

    background-color: #f1f1f1;

    width: 50%;

    margin: 10px;

    text-align: center;

    line-height: 75px;

    font-size: 30px;

}

The only problem with Flexbox is that it does not work in Internet Explorer 10 or earlier versions.

## Navigation Menu

Use float with a list of hyperlinks to create a horizontal menu:

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    overflow: hidden;

    background-color: #333;

}

li {

    float: left;

}

li a {

    display: inline-block;

    color: white;

    text-align: center;

    padding: 14px 16px;

    text-decoration: none;

}

li a:hover {

    background-color: #111;

}

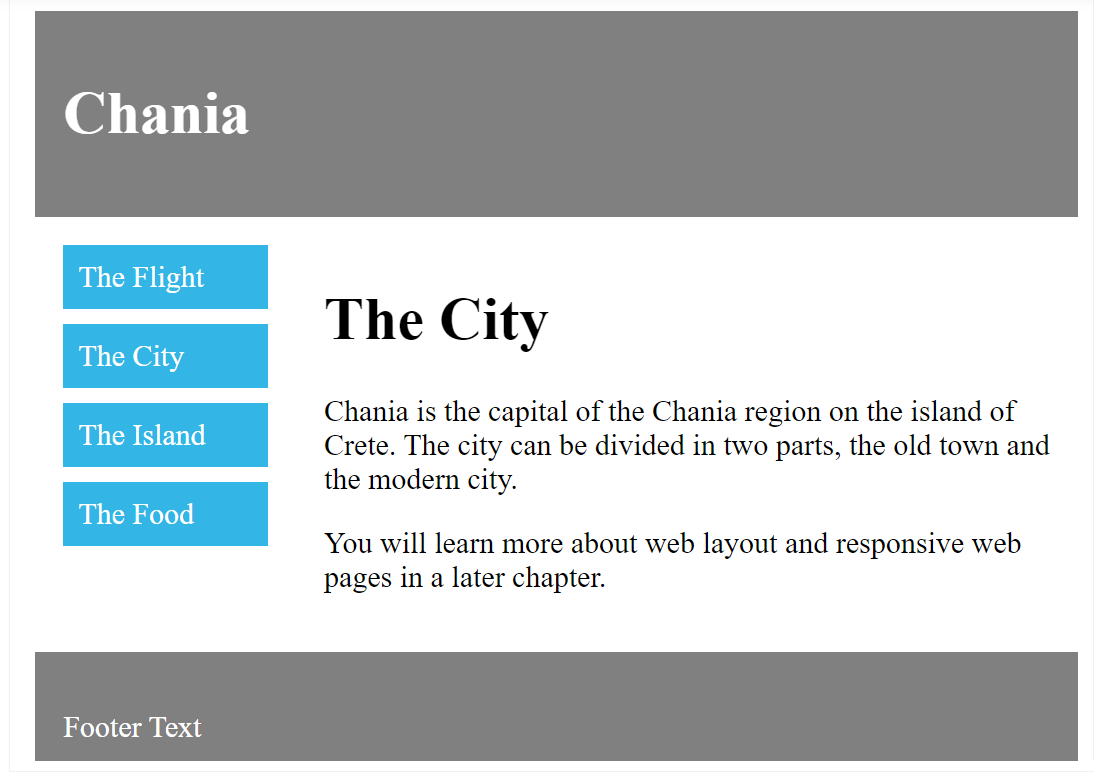
.active {

    background-color: red;

}

## Web Layout Example

It is also common to do entire web layouts using the float property:



\* {

    box-sizing: border-box;

}

.header,

.footer {

    background-color: grey;

    color: white;

    padding: 15px;

}

.column {

    float: left;

    padding: 15px;

}

.clearfix::after {

    content: "";

    clear: both;

    display: table;

}

.menu {

    width: 25%;

}

.content {

    width: 75%;

}

.menu ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

}

.menu li {

    padding: 8px;

    margin-bottom: 8px;

    background-color: #33b5e5;

    color: #ffffff;

}

.menu li:hover {

    background-color: #0099cc;

}

# CSS Layout - display: inline-block

## The display: inline-block Value

Compared to display: inline, the major difference is that display: inline-block allows to set a width and height on the element.

Also, with display: inline-block, the top and bottom margins/paddings are respected, but with display: inline they are not.

Compared to display: block, the major difference is that display: inline-block does not add a line-break after the element, so the element can sit next to other elements.

span.a {

            display: inline;

*/\* the default for span \*/*

            width: 100px;

            height: 100px;

            padding: 5px;

            border: 1px solid blue;

            background-color: yellow;

        }

        span.b {

            display: inline-block;

            width: 100px;

            height: 100px;

            padding: 5px;

            border: 1px solid blue;

            background-color: yellow;

        }

        span.c {

            display: block;

            width: 100px;

            height: 100px;

            padding: 5px;

            border: 1px solid blue;

            background-color: yellow;

        }

## Using inline-block to Create Navigation Links

One common use for display: inline-block is to display list items horizontally instead of vertically.

.nav {

    background-color: yellow;

    list-style-type: none;

    text-align: center;

    margin: 0;

    padding: 0;

}

.nav li {

    display: inline-block;

    font-size: 20px;

    padding: 20px;

}

# CSS Layout - Horizontal & Vertical Align

## Center Align Elements

To horizontally center a block element (like <div>), use margin: auto;

.center {

    margin: auto;

    width: 60%;

    border: 3px solid #73AD21;

    padding: 10px;

}

**Note:** Center aligning has no effect if the width property is not set (or set to 100%).

## Center Align Text

To just center the text inside an element, use text-align: center;

.center {

    text-align: center;

    border: 3px solid green;

  }

## Center an Image

To center an image, set left and right margin to auto and make it into a block element:

img {

    display: block;

    margin-left: auto;

    margin-right: auto;

}

## Left and Right Align - Using position

One method for aligning elements is to use position: absolute;:

.right {

    position: absolute;

    right: 0px;

    width: 300px;

    border: 3px solid #73AD21;

    padding: 10px;

}

**Note:** Absolute positioned elements are removed from the normal flow, and can overlap elements.

## Left and Right Align - Using float

Another method for aligning elements is to use the float property:

.right {

    float: right;

    width: 300px;

    border: 3px solid #73AD21;

    padding: 10px;

}

**Note:** If an element is taller than the element containing it, and it is floated, it will overflow outside of its container. You can use the "**clearfix**" hack to fix this (see example below).

## Center Vertically - Using padding

There are many ways to center an element vertically in CSS. A simple solution is to use top and bottom padding:

.center {

    padding: 70px 0;

    border: 3px solid green;

}

To center both vertically and horizontally, use padding and text-align: center:

## Center Vertically - Using line-height

Another trick is to use the line-height property with a value that is equal to the height property:

.center {

    line-height: 200px;

    height: 200px;

    border: 3px solid green;

    text-align: center;

}

.center p {

    line-height: 1.5;

    display: inline-block;

    vertical-align: middle;

}

## Center Vertically - Using position & transform

If padding and line-height are not options, another solution is to use positioning and the transform property:

.center {

    height: 200px;

    position: relative;

    border: 3px solid green;

}

.center p {

    margin: 0;

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

}

## Center Vertically - Using Flexbox

You can also use flexbox to center things.

.center {

    display: flex;

    justify-content: center;

    align-items: center;

    height: 200px;

    border: 3px solid green;

}

## CSS Combinators

A combinator is something that explains the relationship between the selectors.

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 (~)

## Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

div p {

    background-color: yellow;

}

## Child Selector (>)

The child selector selects all elements that are the children of a specified element.

div>p {

    background-color: yellow;

}

## Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

div+p {

    background-color: yellow;

}

## General Sibling Selector (~)

The general sibling selector selects all elements that are siblings of a specified element.

div~p {

    background-color: yellow;

}

## What are Pseudo-classes?

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

## Anchor Pseudo-classes

Links can be displayed in different ways:

*/\* unvisited link \*/*

a:link {

    color: red;

}

*/\* visited link \*/*

a:visited {

    color: green;

}

*/\* mouse over link \*/*

a:hover {

    color: hotpink;

}

*/\* selected link \*/*

a:active {

    color: blue;

}

**Note:** a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective! a:active MUST come after a:hover in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

## Pseudo-classes and CSS Classes

Pseudo-classes can be combined with CSS classes:

a.highlight:hover {

    color: #ff0000;

}

## Hover on <div>

An example of using the :hover pseudo-class on a <div> element:

div:hover {

    background-color: blue;

}

## Simple Tooltip Hover

Hover over a <div> element to show a <p> element (like a tooltip):

p {

    display: none;

    background-color: yellow;

    padding: 20px;

}

div:hover p {

    display: block;

}

## CSS - The :first-child Pseudo-class

The :first-child pseudo-class matches a specified element that is the first child of another element.

p:first-child {

    color: blue;

}

## Match the first <i> element in all <p> elements

In the following example, the selector matches the first <i> element in all <p> elements:

p i:first-child {

    color: blue;

}

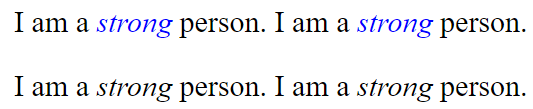
## 

## Match all <i> elements in all first child <p> elements

p:first-child i {

    color: blue;

}



## CSS - The :lang Pseudo-class

The :lang pseudo-class allows you to define special rules for different languages.

q:lang(no) {

    quotes: "~" "~";

}



## What are Pseudo-Elements?

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

## The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

*/\* The following example formats the first line of the text in all <p> elements: \*/*

p::first-line {

    color: #ff0000;

    font-variant: small-caps;

}

**Note:** The ::first-line pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-line pseudo-element:

* font properties
* color properties
* background properties
* word-spacing
* letter-spacing
* text-decoration
* vertical-align
* text-transform
* line-height
* clear

**Notice the double colon notation -**::first-line versus :first-line  
  
The double colon replaced the single-colon notation for pseudo-elements in CSS3. This was an attempt from W3C to distinguish between **pseudo-classes** and **pseudo-elements**.  
  
The single-colon syntax was used for both pseudo-classes and pseudo-elements in CSS2 and CSS1.  
  
For backward compatibility, the single-colon syntax is acceptable for CSS2 and CSS1 pseudo-elements.

## The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

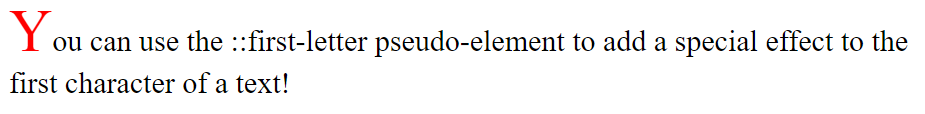
*/\* The following example formats the first letter of the text in all <p> elements:  \*/*

p::first-letter {

    color: #ff0000;

    font-size: xx-large;

}



**Note:** The ::first-letter pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-letter pseudo- element:

* font properties
* color properties
* background properties
* margin properties
* padding properties
* border properties
* text-decoration
* vertical-align (only if "float" is "none")
* text-transform
* line-height
* float
* clear

## Pseudo-elements and CSS Classes

Pseudo-elements can be combined with CSS classes:

p.intro::first-letter {

    color: #ff0000;

    font-size: 200%;

}

## Multiple Pseudo-elements

Several pseudo-elements can also be combined.

*/\* In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color: \*/*

p::first-letter {

    color: #ff0000;

    font-size: xx-large;

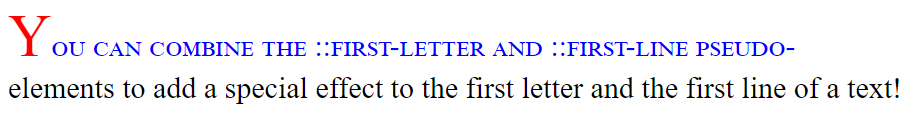
  }

  p::first-line {

    color: #0000ff;

    font-variant: small-caps;

  }



## CSS - The ::before Pseudo-element

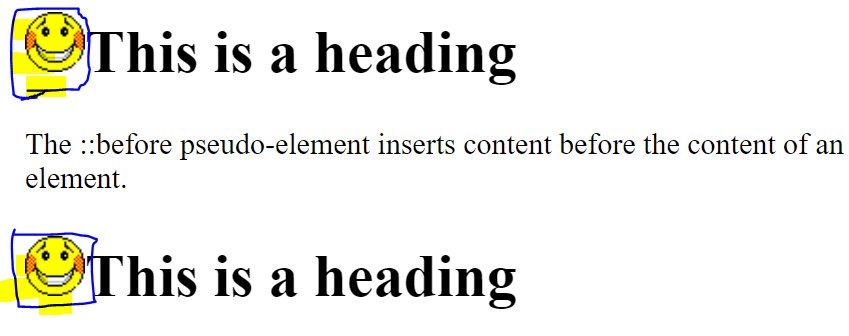
The ::before pseudo-element can be used to insert some content before the content of an element.

*/\* The following example inserts an image before the content of each <h1> element: \*/*

h1::before {

    content: url(smiley.gif);

}



## CSS - The ::after Pseudo-element

The ::after pseudo-element can be used to insert some content after the content of an element.

h1::after {

    content: url(smiley.gif);

}



## CSS - The ::marker Pseudo-element

The ::marker pseudo-element selects the markers of list items.

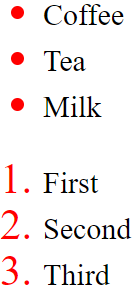
*/\* The following example styles the markers of list items: \*/*

::marker {

    color: red;

    font-size: 23px;

}



## CSS - The ::selection Pseudo-element

The ::selection pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to ::selection: color, background, cursor, and outline.

*/\* The following example makes the selected text red on a yellow background: \*/*

::-moz-selection { */\* Code for Firefox \*/*

    color: red;

    background: yellow;

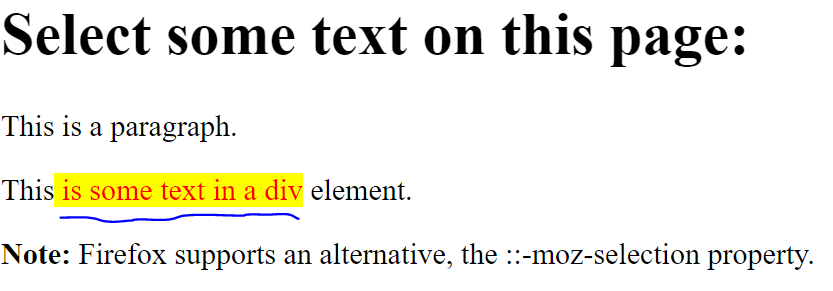
  }

  ::selection {

    color: red;

    background: yellow;

  }



# CSS Opacity / Transparency

The opacity property specifies the opacity/transparency of an element.

## Transparent Image

The opacity property can take a value from 0.0 - 1.0. The lower value, the more transparent:

## Transparent Hover Effect

The opacity property is often used together with the :hover selector to change the opacity on mouse-over:

img {

    opacity: 0.5;

}

img:hover {

    opacity: 1.0;

}

## Transparent Box

When using the opacity property to add transparency to the background of an element, all of its child elements inherit the same transparency. This can make the text inside a fully transparent element hard to read:

div {

    background-color: #4CAF50;

    padding: 10px;

}

div.first {

    opacity: 0.1;

}

div.second {

    opacity: 0.3;

}

div.third {

    opacity: 0.6;

}



## Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:

div {

    background: rgb(76, 175, 80);

    padding: 10px;

}

div.first {

    background: rgba(76, 175, 80, 0.1);

}

div.second {

    background: rgba(76, 175, 80, 0.3);

}

div.third {

    background: rgba(76, 175, 80, 0.6);

}



# CSS Navigation Bar

## Navigation Bar = List of Links

A navigation bar needs standard HTML as a base.

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

}

Example explained:

* 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

# CSS Vertical Navigation Bar

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    width: 200px;

    background-color: #f1f1f1;

}

li a {

    display: block;

    color: #000;

    padding: 8px 16px;

    text-decoration: none;

}

*/\* Change the link color on hover \*/*

li a:hover {

    background-color: #555;

    color: white;

}

### Active/Current Navigation Link

.active {

    background-color: #4CAF50;

    color: white;

}

### Full-height Fixed Vertical Navbar

Create a full-height, "sticky" side navigation:

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    width: 25%;

    background-color: #f1f1f1;

    position: fixed;

    height: 100%;

    overflow: auto;

}

li a {

    display: block;

    color: #000;

    padding: 8px 16px;

    text-decoration: none;

}

li a.active {

    background-color: #4CAF50;

    color: white;

}

li a:hover:not(.active) {

    background-color: #555;

    color: white;

}

# Horizontal Navigation Bar

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

* display: inline; - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

}

li {

    display: inline;

}

Another way of creating a horizontal navigation bar is to float the <li> elements, and specify a layout for the navigation links:

ul {

            list-style-type: none;

            margin: 0;

            padding: 0;

            overflow: hidden;

        }

        li {

            float: left;

        }

        li a {

            display: block;

            padding: 8px;

            background-color: #dddddd;

        }

* float: left; - use float to get block elements to slide next to each other
* display: block; - Allows us to specify padding (and height, width, margins, etc. if you want)

## Horizontal Navigation Bar Examples

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    overflow: hidden;

    background-color: #333;

}

li {

    float: left;

}

li a {

    display: block;

    color: white;

    text-align: center;

    padding: 14px 16px;

    text-decoration: none;

}

li a:hover {

    background-color: #111;

}

### Active/Current Navigation Link

.active {

    background-color: #4CAF50;

  }

### Right-Align Links

Right-align links by floating the list items to the right (float:right;):

ul {

    list-style-type: none;

    margin: 0;

    padding: 0;

    overflow: hidden;

    background-color: #333;

}

li {

    float: left;

}

li a {

    display: block;

    color: white;

    text-align: center;

    padding: 14px 16px;

    text-decoration: none;

}

li a:hover:not(.active) {

    background-color: #111;

}

.active {

    background-color: #4CAF50;

}

HTML

<ul>

        <li><a *href*="#home">Home</a></li>

        <li><a *href*="#news">News</a></li>

        <li><a *href*="#contact">Contact</a></li>

        <li *style*="float:right"><a *class*="active" *href*="#about">About</a></li>

    </ul>

### Border Dividers

Add the border-right property to <li> to create link dividers:

li {

    float: left;

    border-right: 1px solid #bbb;

}

li:last-child {

    border-right: none;

}

### Fixed Navigation Bar

ul {

    position: fixed;

    top: 0;

    width: 100%;

}

body {

            margin: 0;

        }

        ul {

            list-style-type: none;

            margin: 0;

            padding: 0;

            overflow: hidden;

            background-color: #333;

            position: fixed;

            top: 0;

            width: 100%;

        }

        li {

            float: left;

        }

        li a {

            display: block;

            color: white;

            text-align: center;

            padding: 14px 16px;

            text-decoration: none;

        }

        li a:hover:not(.active) {

            background-color: #111;

        }

        .active {

            background-color: #4CAF50;

        }

### Sticky Navbar

Add position: sticky; to <ul> to create a sticky navbar.

ul {

    position: -webkit-sticky; */\* Safari \*/*

    position: sticky;

    top: 0;

  }

# Attribute Selectors

It is possible to style HTML elements that have specific attributes or attribute values.

## CSS [attribute] Selector

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

a[target] {

    background-color: yellow;

}

<a href="http://www.disney.com" target="\_blank">disney.com</a>

## CSS [attribute="value"] Selector

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

a[target=\_blank] {

    background-color: yellow;

}

 <a *href*="http://www.disney.com" *target*="\_blank">disney.com</a>

## CSS [attribute~="value"] Selector

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

[title~=flower] {

    border: 5px solid yellow;

}

<img *src*="klematis.jpg" *title*="**klematis flower**" *width*="150" *height*="113">

<img *src*="img\_flwr.gif" *title*="**flower**" *width*="224" *height*="162">

<img *src*="img\_tree.gif" *title*="**tree**" *width*="200" *height*="358">



## CSS [attribute|="value"] Selector

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

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

[class|=top] {

    background: yellow;

}

<h1 *class*="top-header">Welcome</h1>

<p *class*="top">Hello world!</p>

<p *class*="topcontent">Are you learning CSS?</p>



## CSS [attribute^="value"] Selector

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

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

[class^=top] {

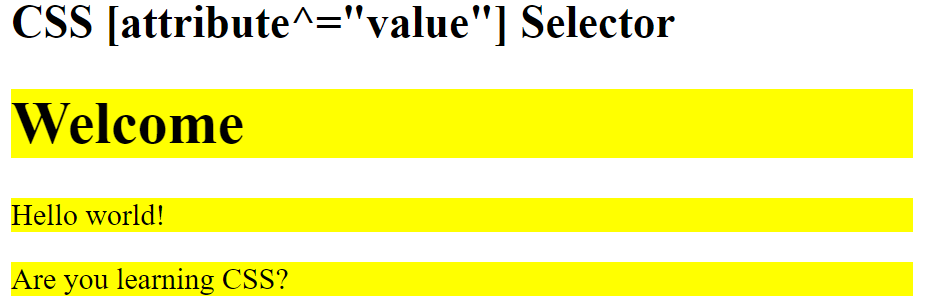
    background: yellow;

}

<h1 *class*="top-header">Welcome</h1>

<p *class*="top">Hello world!</p>

<p *class*="topcontent">Are you learning CSS?</p>



## CSS [attribute$="value"] Selector

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

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

[class$=top] {

    background: yellow;

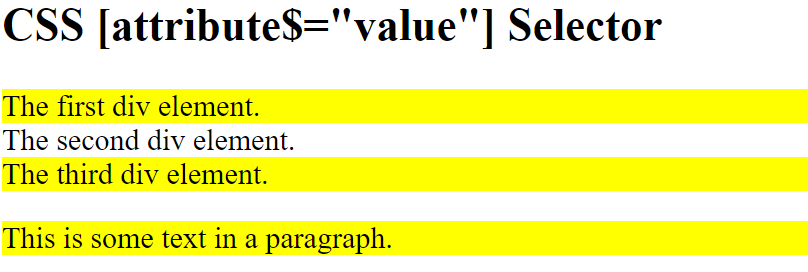
}

<div *class*="first\_test">The first div element.</div>

<div *class*="second">The second div element.</div>

<div *class*="my-test">The third div element.</div>

<p *class*="mytest">This is some text in a paragraph.</p>



## CSS [attribute\*="value"] Selector

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

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

[class\*="te"] {

    background: yellow;

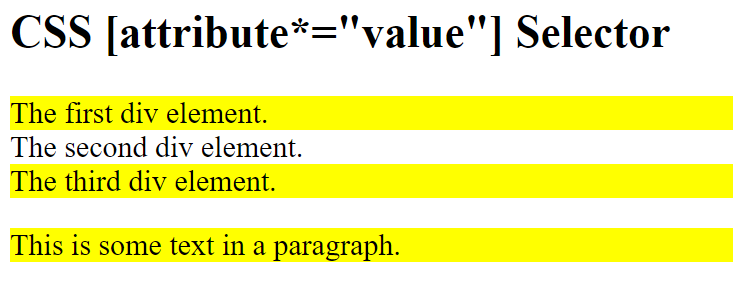
}

<div *class*="first\_test">The first div element.</div>

<div *class*="second">The second div element.</div>

<div *class*="my-test">The third div element.</div>

<p *class*="mytest">This is some text in a paragraph.</p>



## CSS Units

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.

## Relative Lengths

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

## What is !important?

The !important rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the !important rule, it will override ALL previous styling rules for that specific property on that element!

# CSS Rounded Corners

With the CSS border-radius property, you can give any element "rounded corners".

**Tip:** This property allows you to add rounded corners to elements!

#rcorners1 {

    border-radius: 25px;

    background: #73AD21;

    padding: 20px;

    width: 200px;

    height: 150px;

}

#rcorners2 {

    border-radius: 25px;

    border: 2px solid #73AD21;

    padding: 20px;

    width: 200px;

    height: 150px;

}

#rcorners3 {

    border-radius: 25px;

    background: url(paper.gif);

    background-position: left top;

    background-repeat: repeat;

    padding: 20px;

    width: 200px;

    height: 150px;

}

**Tip:** The border-radius property is actually a shorthand property for the border-top-left-radius, border-top-right-radius, border-bottom-right-radius and border-bottom-left-radius properties.

## CSS border-radius - Specify Each Corner

The border-radius property can have from one to four values. Here are the rules:

**Four values - border-radius: 15px 50px 30px 5px;**



**Three values - border-radius: 15px 50px 30px;**



**Two values - border-radius: 15px 50px;**



**One value - border-radius: 15px;**



#rcorners1 {

**border-radius: 50px / 15px;**

    background: #73AD21;

    padding: 20px;

    width: 200px;

    height: 150px;

}



#rcorners2 {

    border-radius: 15px / 50px;

    background: #73AD21;

    padding: 20px;

    width: 200px;

    height: 150px;

}



#rcorners3 {

    border-radius: 50%;

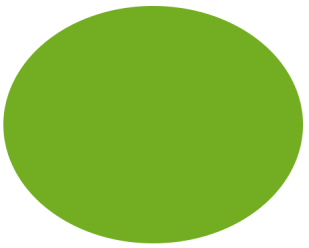
    background: #73AD21;

    padding: 20px;

    width: 200px;

    height: 150px;

}



# CSS Border Images

With the CSS border-image property, you can set an image to be used as the border around an element.

The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.

1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

The border-image property takes the image and slices it into nine sections, like a tic-tac-toe board.

**Note:** For border-image to work, the element also needs the border property set!

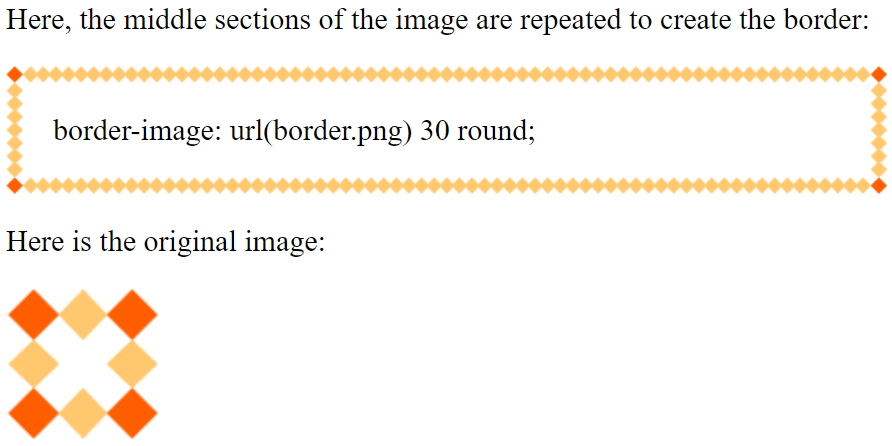
#borderimg {

    border: 10px solid transparent;

    padding: 15px;

    border-image: url(border.png) 30 round;

}



#borderimg {

    border: 10px solid transparent;

    padding: 15px;

**border-image: url(border.png) 30 stretch;**

}



**Tip:** The border-image property is actually a shorthand property for the border-image-source, border-image-slice, border-image-width, border-image-outset and border-image-repeat properties.

## CSS border-image - Different Slice Values

Different slice values completely changes the look of the border:

#borderimg1 {

    border: 10px solid transparent;

    padding: 15px;

    border-image: url(border.png) 50 round;

}

#borderimg2 {

    border: 10px solid transparent;

    padding: 15px;

    border-image: url(border.png) 20% round;

}

#borderimg3 {

    border: 10px solid transparent;

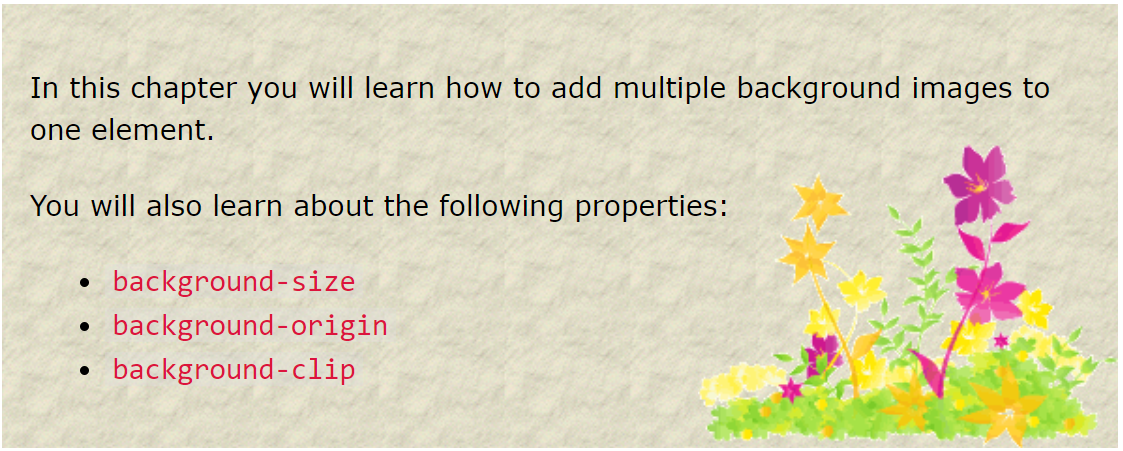
    padding: 15px;

    border-image: url(border.png) 30% round;

}



# CSS Multiple Backgrounds



## CSS Multiple Backgrounds

CSS allows you to add multiple background images for an element, through the background-image property.

The different background images are separated by commas, and the images are stacked on top of each other, where the first image is closest to the viewer.

#example1 {

    background-image: url(img\_flwr.gif), url(paper.gif);

    background-position: right bottom, left top;

    background-repeat: no-repeat, repeat;

    padding: 15px;

}

background shorthand property (same result as example above):

#example1 {

    background: url(img\_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;

    padding: 15px;

}

## CSS Background Size

The CSS background-size property allows you to specify the size of background images.

The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

#example1 {

    border: 1px solid black;

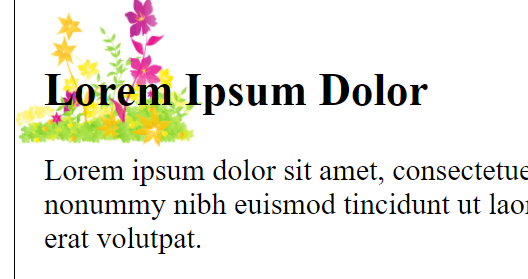
    background: url(img\_flwr.gif);

    background-size: 100px 80px;

    background-repeat: no-repeat;

    padding: 15px;

}



The two other possible values for background-size are contain and cover.

The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area).

The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area).

.div1 {

    border: 1px solid black;

    height: 120px;

    width: 150px;

    background: url(img\_flwr.gif);

    background-repeat: no-repeat;

    background-size: contain;

}

.div2 {

    border: 1px solid black;

    height: 120px;

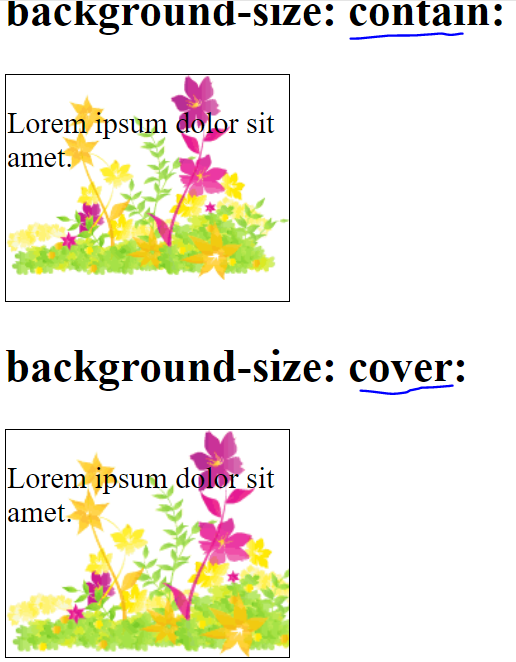
    width: 150px;

    background: url(img\_flwr.gif);

    background-repeat: no-repeat;

    background-size: cover;

}



## Define Sizes of Multiple Background Images

The background-size property also accepts multiple values for background size (using a comma-separated list), when working with multiple backgrounds.

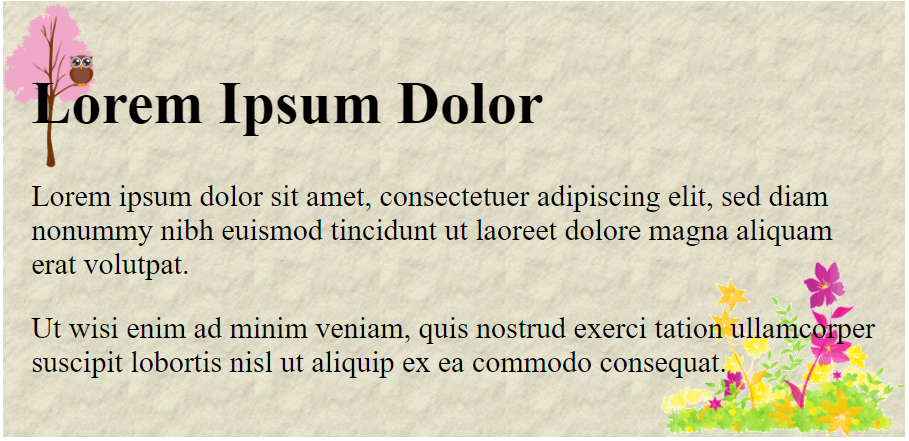
#example1 {

    background: url(img\_tree.gif) left top no-repeat, url(img\_flwr.gif) right bottom no-repeat, url(paper.gif) left top repeat;

    padding: 15px;

    background-size: 50px, 130px, auto;

}



## Full Size Background Image

Now we want to have a background image on a website that covers the entire browser window at all times.

The requirements are as follows:

* Fill the entire page with the image (no white space)
* Scale image as needed
* Center image on page
* Do not cause scrollbars

html {

    background: url(img\_man.jpg) no-repeat center fixed;

    background-size: cover;

}

## Hero Image

You could also use different background properties on a <div> to create a hero image (a large image with text), and place it where you want.

.hero-image {

    background: url(img\_man.jpg) no-repeat center;

    background-size: cover;

    height: 500px;

    position: relative;

}

.hero-text {

    text-align: center;

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

    color: white;

}

Html

<div *class*="hero-image">

        <div *class*="hero-text">

            <h1 *style*="font-size:50px">I am John Doe</h1>

            <h3>And I'm a Photographer</h3>

            <button>Hire me</button>

        </div>

    </div>

## CSS background-origin Property

The CSS background-origin property specifies where the background image is positioned.

The property takes three different values:

* border-box - the background image starts from the upper left corner of the border
* padding-box - (default) the background image starts from the upper left corner of the padding edge
* content-box - the background image starts from the upper left corner of the content

#example1 {

    border: 10px solid black;

    padding: 35px;

    background: url(img\_flwr.gif);

    background-repeat: no-repeat;

    background-origin: border-box;

}

#example2 {

    border: 10px solid black;

    padding: 35px;

    background: url(img\_flwr.gif);

    background-repeat: no-repeat;

    background-origin: border-box;

}

#example3 {

    border: 10px solid black;

    padding: 35px;

    background: url(img\_flwr.gif);

    background-repeat: no-repeat;

    background-origin: content-box;

}

## CSS background-clip Property

The CSS background-clip property specifies the painting area of the background.

#example1 {

    border: 10px dotted black;

    padding: 35px;

    background: yellow;

}

#example2 {

    border: 10px dotted black;

    padding: 35px;

    background: yellow;

    background-clip: padding-box;

}

#example3 {

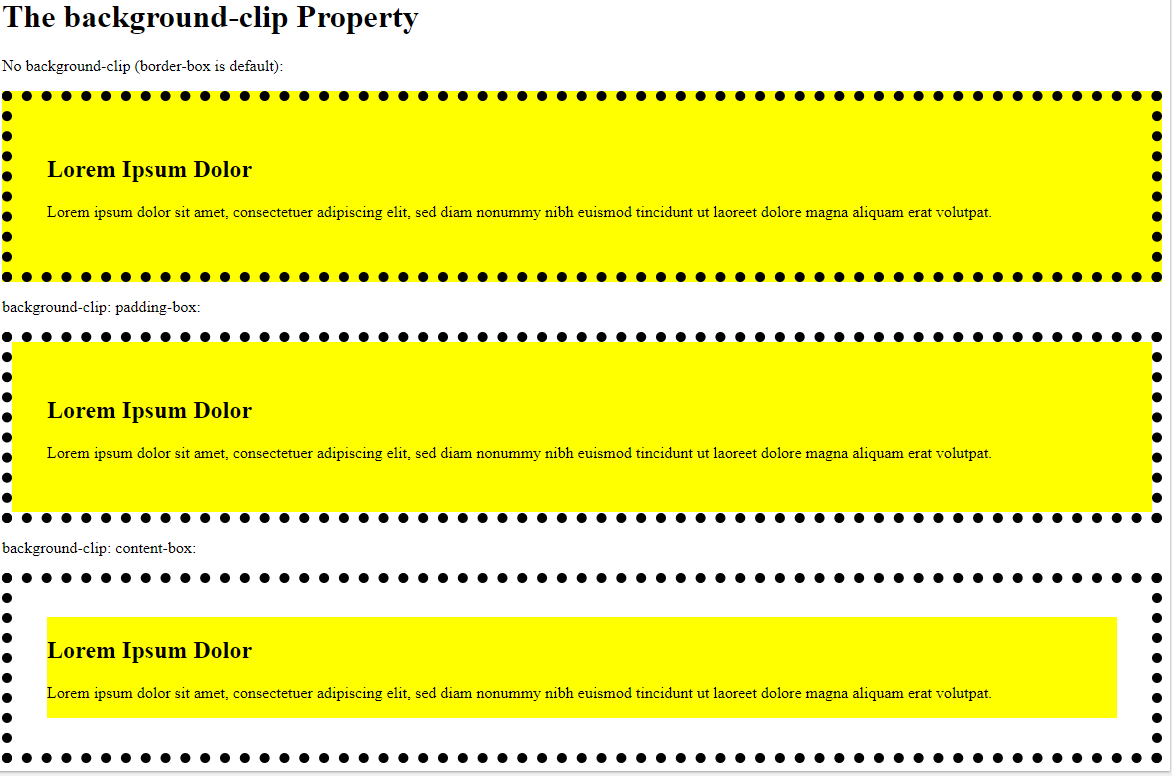
    border: 10px dotted black;

    padding: 35px;

    background: yellow;

    background-clip: content-box;

}



# CSS Color Keywords

This page will explain the transparent, currentcolor, and inherit keywords.

## The transparent Keyword

The transparent keyword is used to make a color transparent.

## The currentcolor Keyword

The currentcolor keyword is like a variable that holds the current value of the color property of an element.

body {

    color: purple;

}

div {

    background-color: currentcolor;

    padding: 15px;

}

## The inherit Keyword

The inherit keyword specifies that a property should inherit its value from its parent element.

The inherit keyword can be used for any CSS property, and on any HTML element.

div {

    border: 2px solid red;

}

span {

    border: inherit;

}



# CSS Gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines two types of gradients:

* **Linear Gradients (goes down/up/left/right/diagonally)**
* **Radial Gradients (defined by their center)**

## CSS Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among.

### Syntax

**background-image: linear-gradient(direction, color-stop1, color-stop2, ...);**

**Direction - Top to Bottom (this is default)**

The following example shows a linear gradient that starts at the top. It starts red, transitioning to yellow:

#grad1 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: linear-gradient(red, yellow);

}



**Direction - Left to Right**

The following example shows a linear gradient that starts from the left. It starts red, transitioning to yellow:

#grad1 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: linear-gradient(to right, red, yellow);

}



**Direction - Diagonal**

You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.

The following example shows a linear gradient that starts at top left (and goes to bottom right). It starts red, transitioning to yellow:

#grad1 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: linear-gradient(to bottom right, red, yellow);

}



## Using Angles

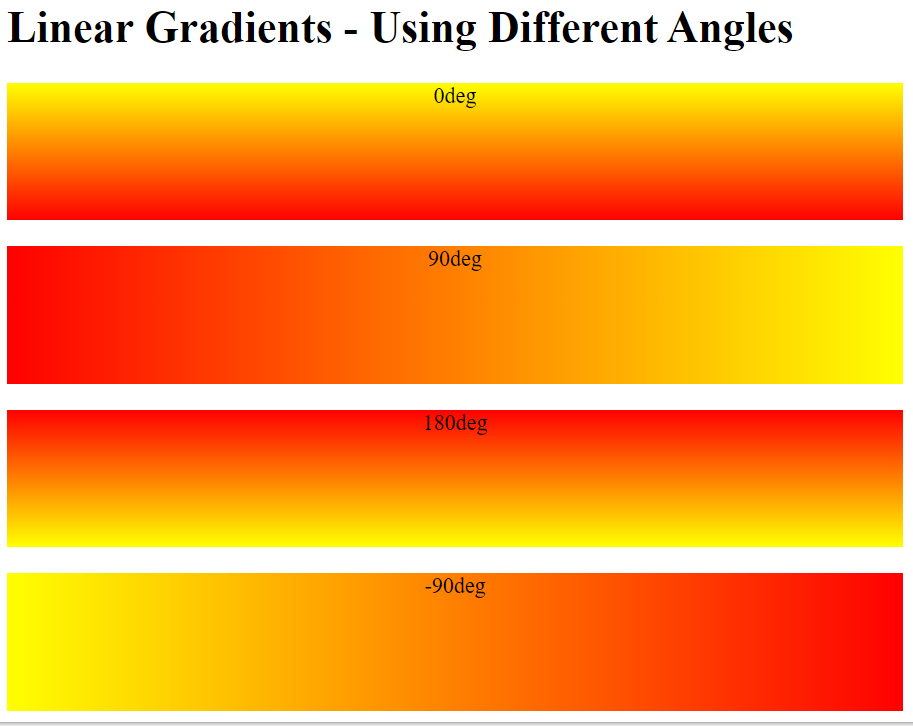
If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.). A value of 0deg is equivalent to "to top". A value of 90deg is equivalent to "to right". A value of 180deg is equivalent to "to bottom".

**background-image: linear-gradient(angle, color-stop1, color-stop2);**

#grad {

    background-image: linear-gradient(180deg, red, yellow);

}

****

## Using Multiple Color Stops

The following example shows a linear gradient (from top to bottom) with multiple color stops:

#grad1 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: linear-gradient(red, yellow, green);

}

#grad2 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: linear-gradient(red, orange, yellow, green, blue, indigo, violet);

}

#grad3 {

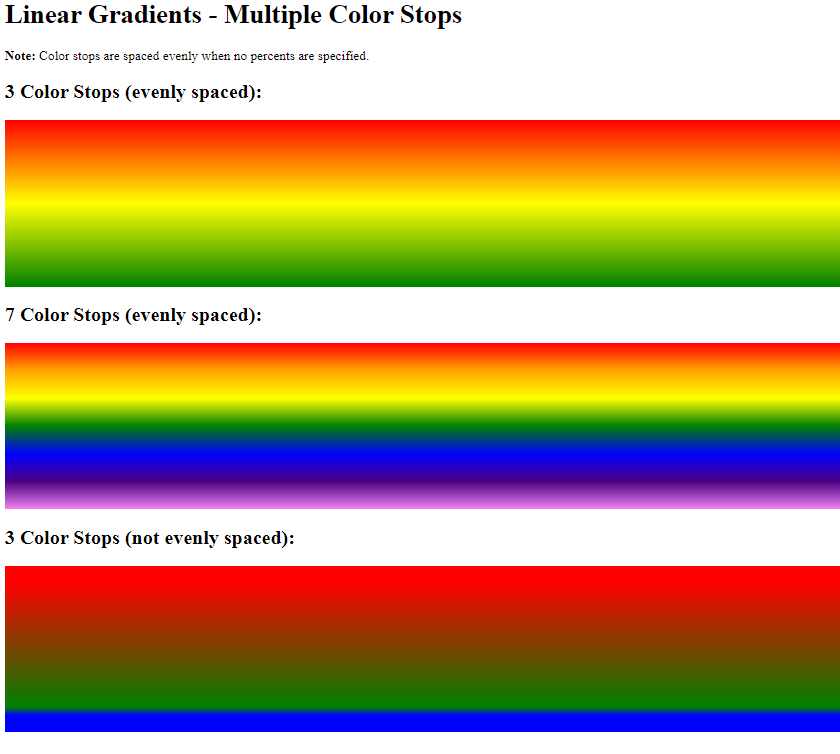
    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: linear-gradient(red 10%, green 85%, blue 90%);

}

****

## Using Transparency

CSS gradients also support transparency, which can be used to create fading effects.

To add transparency, we use the rgba() function to define the color stops. The last parameter in the rgba() function can be a value from 0 to 1, and it defines the transparency of the color: 0 indicates full transparency, 1 indicates full color (no transparency).

#grad1 {

    height: 200px;

    background-image: linear-gradient(to right, rgba(255, 0, 0, 0), rgba(255, 0, 0, 1));

}

****

## Repeating a linear-gradient

The repeating-linear-gradient() function is used to repeat linear gradients:

#grad1 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: repeating-linear-gradient(red, yellow 10%, green 20%);

}

#grad2 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: repeating-linear-gradient(45deg, red, yellow 7%, green 10%);

}

#grad3 {

    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: repeating-linear-gradient(190deg, red, yellow 7%, green 10%);

}

#grad4 {

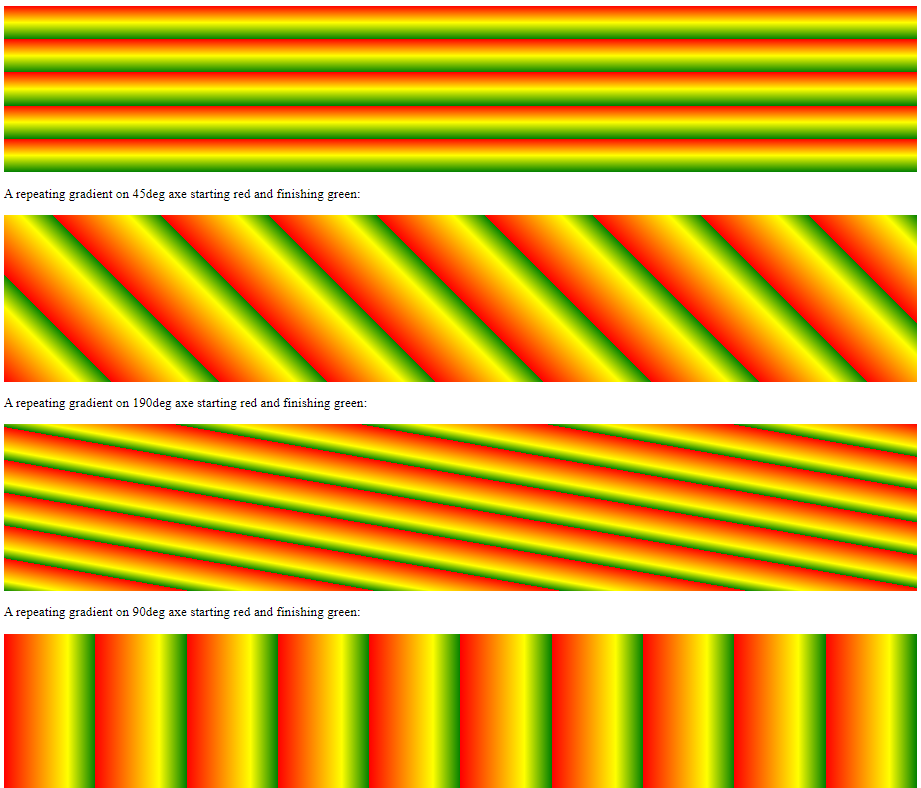
    height: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: repeating-linear-gradient(90deg, red, yellow 7%, green 10%);

}

****

# CSS Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

**background-image: radial-gradient(shape size at position, start-color, ..., last-color);**

By default, shape is ellipse, size is farthest-corner, and position is center.

**Radial Gradient - Evenly Spaced Color Stops (this is default)**

#grad1 {

    height: 150px;

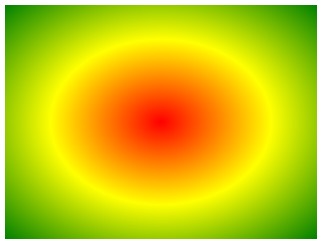
    width: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(red, yellow, green);

}

****

**Radial Gradient - Differently Spaced Color Stops**

#grad1 {

    height: 150px;

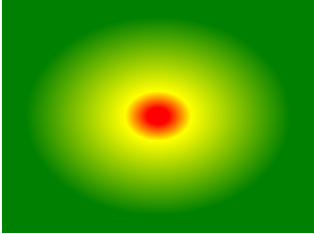
    width: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(red 5%, yellow 15%, green 60%);

}

****

## Set Shape

The shape parameter defines the shape. It can take the value circle or ellipse. The default value is ellipse.

#grad1 {

    height: 150px;

    width: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(red, yellow, green);

}

#grad2 {

    height: 150px;

    width: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(circle, red, yellow, green);

}

****

## Use of Different Size Keywords

The size parameter defines the size of the gradient. It can take four values:

* **closest-side**
* **farthest-side**
* **closest-corner**
* **farthest-corner**

#grad1 {

    height: 150px;

    width: 150px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(closest-side at 60% 55%, red, yellow, black);

}

#grad2 {

    height: 150px;

    width: 150px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(farthest-side at 60% 55%, red, yellow, black);

}

#grad3 {

    height: 150px;

    width: 150px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(closest-corner at 60% 55%, red, yellow, black);

}

#grad4 {

    height: 150px;

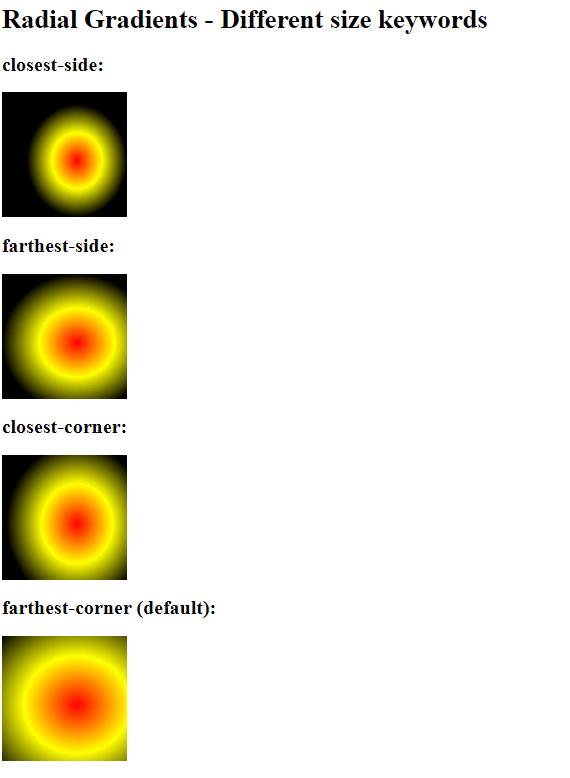
    width: 150px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: radial-gradient(farthest-corner at 60% 55%, red, yellow, black);

}

****

## Repeating a radial-gradient

The repeating-radial-gradient() function is used to repeat radial gradients:

#grad1 {

    height: 150px;

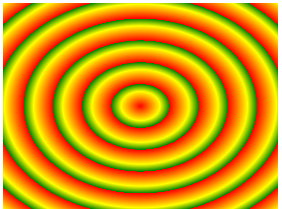
    width: 200px;

    background-color: red;

*/\* For browsers that do not support gradients \*/*

    background-image: repeating-radial-gradient(red, yellow 10%, green 15%);

}

****

# CSS Shadow Effects

With CSS you can add shadow to text and to elements.

* text-shadow
* box-shadow

The CSS text-shadow property applies shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

h1 {

    text-shadow: 2px 2px;

}



Next, add a color to the shadow:

h1 {

    text-shadow: 2px 2px red;

  }



Then, add a blur effect to the shadow:

h1 {

    text-shadow: 2px 2px 5px red;

}



The following example shows a white text with black shadow:

h1 {

    color: white;

    text-shadow: 2px 2px 4px #000000;

}



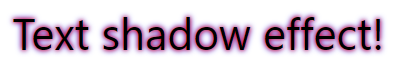
## Multiple Shadows

To add more than one shadow to the text, you can add a comma-separated list of shadows.

h1 {

    text-shadow: 0 0 3px #FF0000, 0 0 5px #0000FF;

}



The following example shows a white text with black, blue, and darkblue shadow:

h1 {

    color: white;

    text-shadow: 1px 1px 2px black, 0 0 25px blue, 0 0 5px darkblue;

}



You can also use the text-shadow property to create a plain border around some text (without shadows):

h1 {

    color: yellow;

    text-shadow: -1px 0 black, 0 1px black, 1px 0 black, 0 -1px black;

}



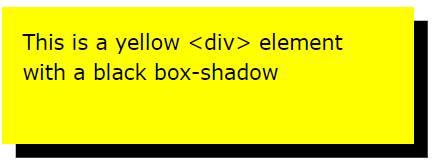
## CSS box-shadow Property

The CSS box-shadow property applies shadow to elements.

div {

    box-shadow: 10px 10px;

}



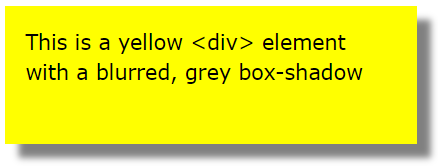
add a color to the shadow:

div {

    box-shadow: 10px 10px grey;

}

 add a blur effect to the shadow:



div {

    box-shadow: 10px 10px 5px grey;

}

|  |  |
| --- | --- |
| **Property** | **Description** |
| [box-shadow](https://www.w3schools.com/cssref/css3_pr_box-shadow.asp) | Adds one or more shadows to an element |
| [text-shadow](https://www.w3schools.com/cssref/css3_pr_text-shadow.asp) | Adds one or more shadows to a text |

# CSS Text Effects

## CSS Text Overflow, Word Wrap, Line Breaking Rules, and Writing Modes

In this chapter you will learn about the following properties:

* text-overflow
* word-wrap
* word-break
* writing-mode

## CSS Text Overflow

The CSS text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.

It can be clipped:



or it can be rendered as an ellipsis (...):

This is some long text that will …

p.test1 {

    white-space: nowrap;

    width: 200px;

    border: 1px solid #000000;

    overflow: hidden;

    text-overflow: clip;

}

p.test2 {

    white-space: nowrap;

    width: 200px;

    border: 1px solid #000000;

    overflow: hidden;

    text-overflow: ellipsis;

}

The following example shows how you can display the overflowed content when hovering over the element:

div.test:hover {

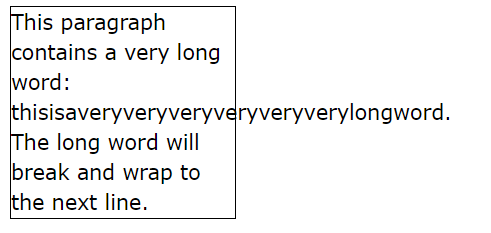
    overflow: visible;

}

## CSS Word Wrapping

The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.

If a word is too long to fit within an area, it expands outside:



The word-wrap property allows you to force the text to wrap - even if it means splitting it in the middle of a word:

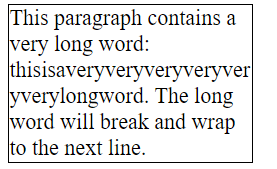
p.test {

    width: 11em;

    border: 1px solid #000000;

    word-wrap: break-word;

}



## CSS Word Breaking

The CSS word-break property specifies line breaking rules.

p.test1 {

    width: 140px;

    border: 1px solid #000000;

    word-break: keep-all;    */\*This line will-break-at-hyphens. \*/*

}

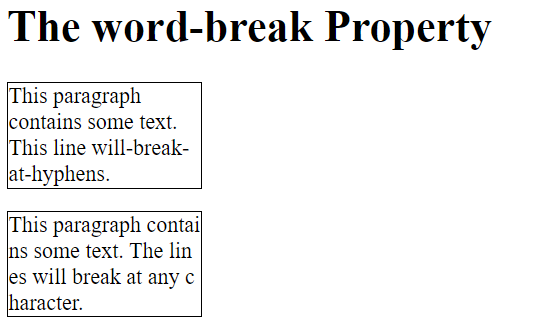
p.test2 {

    width: 140px;

    border: 1px solid #000000;

    word-break: break-all;    */\*The lines will break at any character.\*/*

}



## CSS Writing Mode

The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically.

Some text with a span element with a vertical-rl writing-mode.

p.test1 {

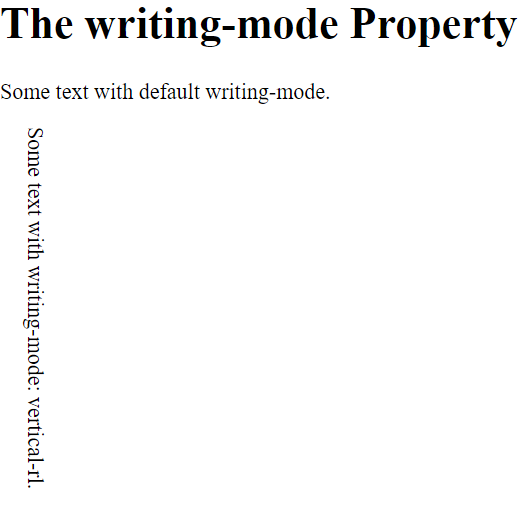
    writing-mode: horizontal-tb;

}

p.test2 {

    writing-mode: vertical-rl;

}



## CSS Text Effect Properties

The following table lists the CSS text effect properties:

|  |  |
| --- | --- |
| **Property** | **Description** |
| [text-align-last](https://www.w3schools.com/cssref/css3_pr_text-align-last.asp) | Specifies how to align the last line of a text |
| [text-justify](https://www.w3schools.com/cssref/css3_pr_text-justify.asp) | Specifies how justified text should be aligned and spaced |
| [text-overflow](https://www.w3schools.com/cssref/css3_pr_text-overflow.asp) | Specifies how overflowed content that is not displayed should be signaled to the user |
| [word-break](https://www.w3schools.com/cssref/css3_pr_word-break.asp) | Specifies line breaking rules for non-CJK scripts |
| [word-wrap](https://www.w3schools.com/cssref/css3_pr_word-wrap.asp) | Allows long words to be able to be broken and wrap onto the next line |
| [writing-mode](https://www.w3schools.com/cssref/css3_pr_writing-mode.asp) | Specifies whether lines of text are laid out horizontally or vertically |

## The CSS @font-face Rule

Web fonts allow Web designers to use fonts that are not installed on the user's computer.

When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed.

Your "own" fonts are defined within the CSS @font-face rule.

## Different Font Formats

**TrueType Fonts (TTF)**

TrueType is a font standard developed in the late 1980s, by Apple and Microsoft. TrueType is the most common font format for both the Mac OS and Microsoft Windows operating systems.

**OpenType Fonts (OTF)**

OpenType is a format for scalable computer fonts. It was built on TrueType, and is a registered trademark of Microsoft. OpenType fonts are used commonly today on the major computer platforms.

**The Web Open Font Format (WOFF)**

WOFF is a font format for use in web pages. It was developed in 2009, and is now a W3C Recommendation. WOFF is essentially OpenType or TrueType with compression and additional metadata. The goal is to support font distribution from a server to a client over a network with bandwidth constraints.

**The Web Open Font Format (WOFF 2.0)**

TrueType/OpenType font that provides better compression than WOFF 1.0.

**SVG Fonts/Shapes**

SVG fonts allow SVG to be used as glyphs when displaying text. The SVG 1.1 specification define a font module that allows the creation of fonts within an SVG document. You can also apply CSS to SVG documents, and the @font-face rule can be applied to text in SVG documents.

**Embedded OpenType Fonts (EOT)**

EOT fonts are a compact form of OpenType fonts designed by Microsoft for use as embedded fonts on web pages.

## Using The Font You Want

In the @font-face rule; first define a name for the font (e.g. myFirstFont) and then point to the font file.

**Tip:** Always use lowercase letters for the font URL. Uppercase letters can give unexpected results in IE.

@font-face {

    font-family: myFirstFont;

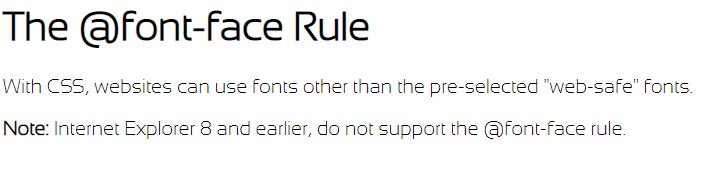
    src: url(sansation\_light.woff);

}

div {

    font-family: myFirstFont;

}



## Using Bold Text

You must add another @font-face rule containing descriptors for bold text:

@font-face {

    font-family: myFirstFont;

    src: url(sansation\_bold.woff);

    font-weight: bold;

  }

## CSS 2D Transforms

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

* transform

## CSS 2D Transforms Methods

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

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

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

The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

div {

    transform: translate(50px, 100px);

}

## The rotate() Method

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

div {

    width: 300px;

    height: 100px;

    background-color: yellow;

    border: 1px solid black;

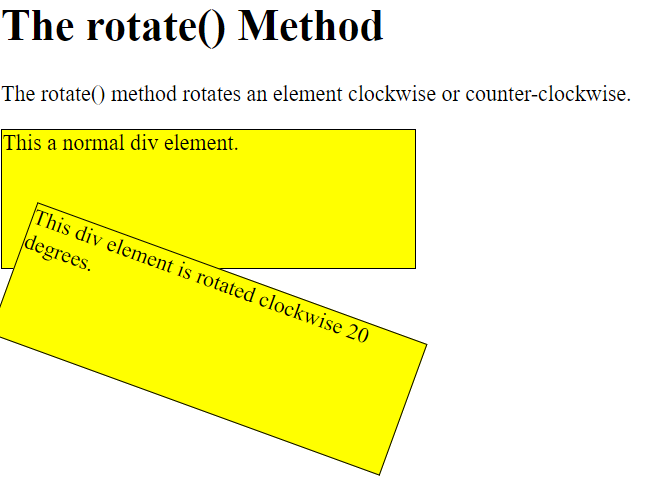
  }

  div#myDiv {

    -ms-transform: rotate(20deg); */\* IE 9 \*/*

    transform: rotate(20deg); */\* Standard syntax \*/*

  }



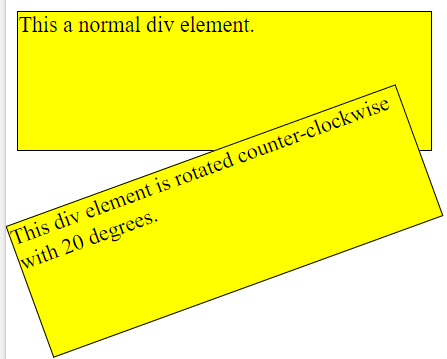
Using negative values will rotate the element counter-clockwise.

div#myDiv {

    -ms-transform: rotate(-20deg); */\* IE 9 \*/*

    transform: rotate(-20deg); */\* Standard syntax \*/*

  }



## The scale() Method

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

div {

    margin: 150px;

    width: 200px;

    height: 100px;

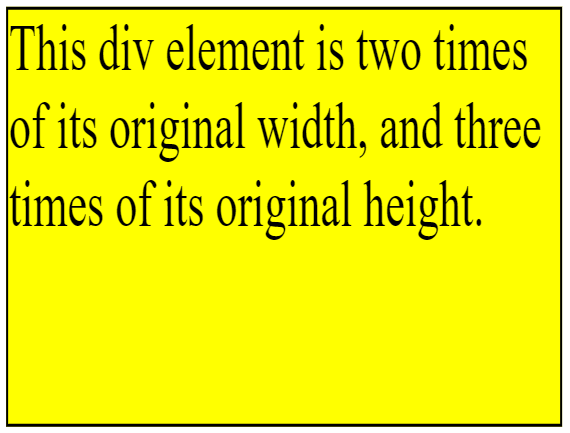
    background-color: yellow;

    border: 1px solid black;

    -ms-transform: scale(2,3); */\* IE 9 \*/*

    transform: scale(2,3); */\* Standard syntax \*/*

  }



The following example decreases the <div> element to be half of its original width and height:

div {

    transform: scale(0.5, 0.5);

}

## The scaleX() Method

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

div {

    transform: scaleX(2);

}

## The scaleY() Method

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

div {

    transform: scaleY(3);

}

## The skewX() Method

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

div {

    width: 300px;

    height: 100px;

    background-color: yellow;

    border: 1px solid black;

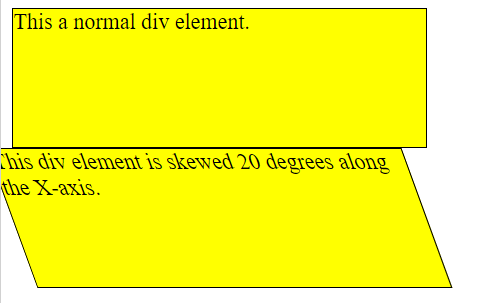
  }

  div#myDiv {

    -ms-transform: skewX(20deg); */\* IE 9 \*/*

    transform: skewX(20deg); */\* Standard syntax \*/*

  }



## The skewY() Method

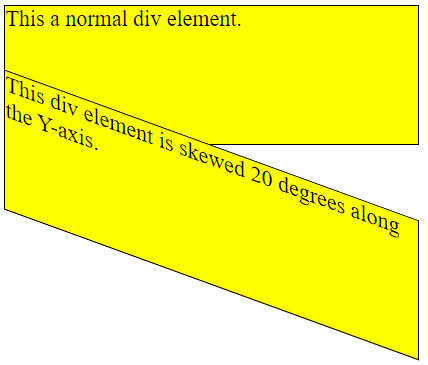
The skewY() method skews an element along the Y-axis by the given angle.

div#myDiv {

    -ms-transform: skewY(20deg); */\* IE 9 \*/*

    transform: skewY(20deg); */\* Standard syntax \*/*

  }



## The skew() Method

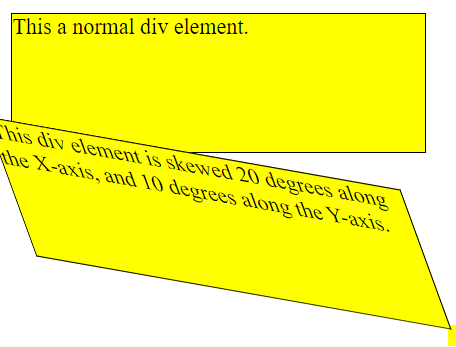
The skew() method skews an element along the X and Y-axis by the given angles.

div#myDiv {

    -ms-transform: skew(20deg,10deg); */\* IE 9 \*/*

    transform: skew(20deg,10deg); */\* Standard syntax \*/*

  }



If the second parameter is not specified, it has a zero value.

## The matrix() Method

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

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

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

div#myDiv1 {

    -ms-transform: matrix(1, -0.3, 0, 1, 0, 0); */\* IE 9 \*/*

    transform: matrix(1, -0.3, 0, 1, 0, 0); */\* Standard syntax \*/*

  }

  div#myDiv2 {

    -ms-transform: matrix(1, 0, 0.5, 1, 150, 0); */\* IE 9 \*/*

    transform: matrix(1, 0, 0.5, 1, 150, 0); */\* Standard syntax \*/*

  }

## CSS 3D Transforms

## CSS 3D Transforms Methods

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

* rotateX()
* rotateY()
* rotateZ()

## The rotateX() Method

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

#myDiv {

    transform: rotateX(150deg);

  }



## The rotateY() Method

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

## The rotateZ() Method

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

## CSS Transitions

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

In this chapter you will learn about the following properties:

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

## How to Use CSS Transitions?

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

div {

    width: 100px;

    height: 100px;

    background: red;

    transition: width 2s;

}

div:hover {

    width: 300px;

}

## Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and height:

div {

    width: 100px;

    height: 100px;

    background: red;

    transition: width 2s, height 2s;

}

div:hover {

    width: 300px;

    height: 300px;

}

## Specify the Speed Curve of the Transition

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

div {

    width: 100px;

    height: 100px;

    background: red;

    transition: width 2s;

  }

  #div1 {transition-timing-function: linear;}

  #div2 {transition-timing-function: ease;}

  #div3 {transition-timing-function: ease-in;}

  #div4 {transition-timing-function: ease-out;}

  #div5 {transition-timing-function: ease-in-out;}

  div:hover {

    width: 300px;

  }

## Delay the Transition Effect

The transition-delay property specifies a delay (in seconds) for the transition effect.

div {

    width: 100px;

    height: 100px;

    background: red;

    transition: width 3s;

    transition-delay: 1s;

  }

  div:hover {

    width: 300px;

  }

## 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);

  }

## More Transition Examples

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

div {

    width: 100px;

    height: 100px;

    background: red;

    transition-property: width;

    transition-duration: 2s;

    transition-timing-function: linear;

    transition-delay: 1s;

  }

  div:hover {

    width: 300px;

  }

## CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

In this chapter you will learn about the following properties:

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