**Information about some important properties gathered from various websites:**

**Meta Tag:**A tag with the**http-equiv** attribute set to **“X-UA-Compatible”** and the content attribute set to **“IE=edge”** is used to force **Internet Explorer (IE)** to use the latest version of the rendering engine. Regardless of the document’s compatibility mode.

When IE detects this meta tag, it renders the web page using the latest rendering engine available on the user’s computer. This ensures that the page is rendered using a newer standards-compliant rendering engine, rather than an older or compatibility-mode engine.

Additionally, the **`IE=edge`**value instructs IE to use the highest mode available, which is typically the most standards-compliant mode. This can help ensure that your website or web application is rendered consistently across different versions of IE and that it takes advantage of the latest web standards and features.

**HTML Viewport meta tag for Responsive Web Design**

**A Browser’s viewport** is the area of the web page in which the content is visible to the user. The viewport does not have the same size, it varies with the variation in screen size of the devices on which the website is visible. For a laptop, the viewport has a larger size as compared to a smartphone or tablet.

**Note:** When a page is not made responsive for smaller viewports it looks bad or even breaks on a smaller screen. To fix this problem introduce a responsive tag to control the viewport. This tag was first introduced by Apple Inc. for Safari iOS.

**Syntax**

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

* The ‘ **width=device-width** ' in meta tag sets the width of the page to follow the screen width of the device, which will vary depending on the device.
* The ‘ **initial-scale=1.0** ‘ in meta tag sets the initial zoom level when the page is first loaded by the browser.

The <header> element represents a container for introductory content or a set of navigational links.

A <header> element typically contains:

* one or more heading elements (<h1> - <h6>)
* logo or icon
* authorship information

**Note:** You can have several <header> elements in one HTML document. However, <header> cannot be placed within a <footer>, <address> or another <header> element.

The <div> tag defines a division or a section in an HTML document.

The <div> tag is used as a container for HTML elements - which is then styled with CSS or manipulated with JavaScript.

The <div> tag is easily styled by using the class or id attribute.

Any sort of content can be put inside the <div> tag!

**Note:** By default, browsers always place a line break before and after the <div> element.

It is called a division tag. The <div> tag is a block-level element that only represents its child elements and doesn’t have a special meaning. It takes the Whole Width available on the screen. It is generally used with the title and class attributes.

The <div> tag is one of the most used tags in website creation. Use <div> element for style purposes or for wrapping paragraphs within a section that are all to be given similar properties. Requires closing </div> tag too.

**HTML section Tag**

The <section> tag is not a generic container in a web-page. The content inside <section> tag will be grouped i.e. it’ll connect to a single subject and appear as an entry in an outline of the page. A common rule is that the <section> element is valid only if that element’s contents would be listed explicitly in the document’s outline.

Section tag is used to distribute the content with a similar theme. The main advantage of the section tag is, it describes its meaning in a web page. It is mostly used when headers, footers, or any other section of documents are needed in a web page. Requires closing </section> tag too.

## CSS Box Sizing

The CSS box-sizing property allows us to include the padding and border in an element's total width and height.

## Without the CSS box-sizing Property

By default, the width and height of an element is calculated like this:

width + padding + border = actual width of an element  
height + padding + border = actual height of an element

This means: When you set the width/height of an element, the element often appears bigger than you have set (because the element's border and padding are added to the element's specified width/height).

The following illustration shows two <div> elements with the same specified width and height:

This div is smaller (width is 300px and height is 100px).

This div is bigger (width is also 300px and height is 100px).

The two <div> elements above end up with different sizes in the result (because div2 has a padding specified):

### **Example**

.div1 {  
  width: 300px;  
  height: 100px;  
  border: 1px solid blue;  
}  
  
.div2 {  
  width: 300px;  
  height: 100px;  
  padding: 50px;  
  border: 1px solid red;  
}

## With the CSS box-sizing Property

The box-sizing property allows us to include the padding and border in an element's total width and height.

If you set box-sizing: border-box; on an element, padding and border are included in the width and height:

Both divs are the same size now!

Hooray!

Here is the same example as above, with box-sizing: border-box; added to both <div> elements:

### **Example**

.div1 {  
  width: 300px;  
  height: 100px;  
  border: 1px solid blue;  
  box-sizing: border-box;  
}  
  
.div2 {  
  width: 300px;  
  height: 100px;  
  padding: 50px;  
  border: 1px solid red;  
  box-sizing: border-box;  
}

CSS Flexbox Layout Module

Before the Flexbox Layout module, there were four layout modes:

* Block, for sections in a webpage
* Inline, for text
* Table, for two-dimensional table data
* Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

## Parent Element (Container)

Like we specified in the previous chapter, this is a flex **container**(the blue area) with three flex **items**:

# 1

# 2

# 3

The flex container becomes flexible by setting the display property to flex:

### **Example**

.flex-container {  
  display: flex;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox)

The flex container properties are:

* [flex-direction](https://www.w3schools.com/css/css3_flexbox_container.asp#flex-direction)
* [flex-wrap](https://www.w3schools.com/css/css3_flexbox_container.asp#flex-wrap)
* [flex-flow](https://www.w3schools.com/css/css3_flexbox_container.asp#flex-flow)
* [justify-content](https://www.w3schools.com/css/css3_flexbox_container.asp#justify-content)
* [align-items](https://www.w3schools.com/css/css3_flexbox_container.asp#align-items)
* [align-content](https://www.w3schools.com/css/css3_flexbox_container.asp#align-content)

ADVERTISEMENT

## The flex-direction Property

The flex-direction property defines in which direction the container wants to stack the flex items.

# 1

# 2

# 3

### **Example**

The column value stacks the flex items vertically (from top to bottom):

.flex-container {  
  display: flex;  
  flex-direction: column;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-direction_column)

### **Example**

The column-reverse value stacks the flex items vertically (but from bottom to top):

.flex-container {  
  display: flex;  
  flex-direction: column-reverse;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-direction_column-reverse)

### **Example**

The row value stacks the flex items horizontally (from left to right):

.flex-container {  
  display: flex;  
  flex-direction: row;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-direction_row)

### **Example**

The row-reverse value stacks the flex items horizontally (but from right to left):

.flex-container {  
  display: flex;  
  flex-direction: row-reverse;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-direction_row-reverse)

## The flex-wrap Property

The flex-wrap property specifies whether the flex items should wrap or not.

The examples below have 12 flex items, to better demonstrate the flex-wrap property.

# 1

# 2

# 3

# 4

# 5

# 6

# 7

# 8

# 9

# 10

# 11

# 12

### **Example**

The wrap value specifies that the flex items will wrap if necessary:

.flex-container {  
  display: flex;  
  flex-wrap: wrap;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-wrap_wrap)

### **Example**

The nowrap value specifies that the flex items will not wrap (this is default):

.flex-container {  
  display: flex;  
  flex-wrap: nowrap;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-wrap_nowrap)

### **Example**

The wrap-reverse value specifies that the flexible items will wrap if necessary, in reverse order:

.flex-container {  
  display: flex;  
  flex-wrap: wrap-reverse;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-wrap_wrap-reverse)

## The flex-flow Property

The flex-flow property is a shorthand property for setting both the flex-direction and flex-wrap properties.

### **Example**

.flex-container {  
  display: flex;  
  flex-flow: row wrap;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_flex-flow_row_wrap)

## The justify-content Property

The justify-content property is used to align the flex items:

# 1

# 2

# 3

### **Example**

The center value aligns the flex items at the center of the container:

.flex-container {  
  display: flex;  
  justify-content: center;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_justify-content_center)

### **Example**

The flex-start value aligns the flex items at the beginning of the container (this is default):

.flex-container {  
  display: flex;  
  justify-content: flex-start;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_justify-content_flex-start)

### **Example**

The flex-end value aligns the flex items at the end of the container:

.flex-container {  
  display: flex;  
  justify-content: flex-end;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_justify-content_flex-end)

### **Example**

The space-around value displays the flex items with space before, between, and after the lines:

.flex-container {  
  display: flex;  
  justify-content: space-around;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_justify-content_space-around)

### **Example**

The space-between value displays the flex items with space between the lines:

.flex-container {  
  display: flex;  
  justify-content: space-between;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_justify-content_space-between)

## The align-items Property

The align-items property is used to align the flex items.

# 1

# 2

# 3

In these examples we use a 200 pixels high container, to better demonstrate the align-items property.

### **Example**

The center value aligns the flex items in the middle of the container:

.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: center;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-items_center)

### **Example**

The flex-start value aligns the flex items at the top of the container:

.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: flex-start;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-items_flex-start)

### **Example**

The flex-end value aligns the flex items at the bottom of the container:

.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: flex-end;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-items_flex-end)

### **Example**

The stretch value stretches the flex items to fill the container (this is default):

.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: stretch;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-items_stretch)

### **Example**

The baseline value aligns the flex items such as their baselines aligns:

.flex-container {  
  display: flex;  
  height: 200px;  
  align-items: baseline;  
}

**Note:** the example uses different font-size to demonstrate that the items gets aligned by the text baseline:

# 1

###### **2**

### **3**

4

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-items_baseline)

## The align-content Property

The align-content property is used to align the flex lines.

# 1

# 2

# 3

# 4

# 5

# 6

# 7

# 8

# 9

# 10

# 11

# 12

In these examples we use a 600 pixels high container, with the flex-wrap property set to wrap, to better demonstrate the align-content property.

### **Example**

The space-between value displays the flex lines with equal space between them:

.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: space-between;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-content_space-between)

### **Example**

The space-around value displays the flex lines with space before, between, and after them:

.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: space-around;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-content_space-around)

### **Example**

The stretch value stretches the flex lines to take up the remaining space (this is default):

.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: stretch;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-content_stretch)

### **Example**

The center value displays the flex lines in the middle of the container:

.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: center;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-content_center)

### **Example**

The flex-start value displays the flex lines at the start of the container:

.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: flex-start;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-content_flex-start)

### **Example**

The flex-end value displays the flex lines at the end of the container:

.flex-container {  
  display: flex;  
  height: 600px;  
  flex-wrap: wrap;  
  align-content: flex-end;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_align-content_flex-end)

## Perfect Centering

In the following example we will solve a very common style problem: perfect centering.

**SOLUTION:** Set both the justify-content and align-items properties to center, and the flex item will be perfectly centered:

### **Example**

.flex-container {  
  display: flex;  
  height: 300px;  
**justify-content: center;  
  align-items: center;**}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_flexbox_perfect_center)

## The CSS Flexbox Container Properties

The following table lists all the CSS Flexbox Container properties:

|  |  |
| --- | --- |
| **Property** | **Description** |
| [align-content](https://www.w3schools.com/cssref/css3_pr_align-content.asp) | Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines |
| [align-items](https://www.w3schools.com/cssref/css3_pr_align-items.asp) | Vertically aligns the flex items when the items do not use all available space on the cross-axis |
| [display](https://www.w3schools.com/cssref/pr_class_display.asp) | Specifies the type of box used for an HTML element |
| [flex-direction](https://www.w3schools.com/cssref/css3_pr_flex-direction.asp) | Specifies the direction of the flexible items inside a flex container |
| [flex-flow](https://www.w3schools.com/cssref/css3_pr_flex-flow.asp) | A shorthand property for flex-direction and flex-wrap |
| [flex-wrap](https://www.w3schools.com/cssref/css3_pr_flex-wrap.asp) | Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line |
| [justify-content](https://www.w3schools.com/cssref/css3_pr_justify-content.asp) | Horizontally aligns the flex items when the items do not use all available space on the main-axis |

An element's padding is the space between its content and its border.

The padding property is a shorthand property for:

* [padding-top](https://www.w3schools.com/cssref/pr_padding-top.php)
* [padding-right](https://www.w3schools.com/cssref/pr_padding-right.php)
* [padding-bottom](https://www.w3schools.com/cssref/pr_padding-bottom.php)
* [padding-left](https://www.w3schools.com/cssref/pr_padding-left.php)

**Note:** Padding creates extra space within an element, while margin creates extra space around an element.

This property can have from one to four values.

**If the padding property has four values:**

* padding:10px 5px 15px 20px;
  + top padding is 10px
  + right padding is 5px
  + bottom padding is 15px
  + left padding is 20px

**If the padding property has three values:**

* padding:10px 5px 15px;
  + top padding is 10px
  + right and left padding are 5px
  + bottom padding is 15px

**If the padding property has two values:**

* padding:10px 5px;
  + top and bottom padding are 10px
  + right and left padding are 5px

**If the padding property has one value:**

* padding:10px;
  + all four paddings are 10px

## CSS Syntax

padding: *length*|initial|inherit;

## Property Values

|  |  |  |
| --- | --- | --- |
| **Value** | **Description** | **Demo** |
| *length* | Specifies the padding in px, pt, cm, etc. Default value is 0. [Read about length units](https://www.w3schools.com/cssref/css_units.php) | [Demo ❯](https://www.w3schools.com/cssref/playdemo.php?filename=playcss_padding) |
| *%* | Specifies the padding in percent of the width of the containing element | [Demo ❯](https://www.w3schools.com/cssref/playdemo.php?filename=playcss_padding&preval=5%25) |
| initial | Sets this property to its default value. [Read about initial](https://www.w3schools.com/cssref/css_initial.php) |  |
| inherit | Inherits this property from its parent element. [Read about inherit](https://www.w3schools.com/cssref/css_inherit.php) |  |

**fas fa-map-marker-alt ----- location symbol**

The CSS **cursor** property determines the appearance of the mouse cursor when hovering over an element to which this property is applied. This property is only applicable in environments with mouse and cursor functionality. Its main purpose is to improve usability by visually representing certain functions.

|  |  |  |
| --- | --- | --- |
| **pointer** |  | The displayed cursor is pointer cursor, showing the cursor serves as an indicator pointing to a hyperlink. |

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

## 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
* 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):
* overflow-x specifies what to do with the left/right edges of the content.  
  overflow-y specifies what to do with the top/bottom edges of the content.
* You can use the overflow property when you want to have better control of the layout. The overflow property

# CSS Outline

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

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

This element has a black border and a green outline with a width of 10px.

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss_outline_intro)

## CSS Outline

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.

ADVERTISEMENT

## 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

The following example shows the different outline-style values:

### **Example**

Demonstration of the different outline styles:

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;}

Result:

A dotted outline.

A dashed outline.

A solid outline.

A double outline.

A groove outline. The effect depends on the outline-color value.

A ridge outline. The effect depends on the outline-color value.

An inset outline. The effect depends on the outline-color value.

An outset outline. The effect depends on the outline-color value.

The height property sets the height of an element.

The height of an element does not include padding, borders, or margins!

If height: auto; the element will automatically adjust its height to allow its content to be displayed correctly.

If height is set to a numeric value (like pixels, (r)em, percentages) then if the content does not fit within the specified height, it will overflow. How the container will handle the overflowing content is defined by the [overflow](https://www.w3schools.com/cssref/pr_pos_overflow.php) property.

**Note:** The [min-height](https://www.w3schools.com/cssref/pr_dim_min-height.php) and [max-height](https://www.w3schools.com/cssref/pr_dim_max-height.php) properties override the height property.

## What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times as you want.

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

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

## The @keyframes Rule

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

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

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

### **Example**

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

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_animation1)

**Note:** The animation-duration property defines how long an animation should take to complete. If the animation-duration property is not specified, no animation will occur, because the default value is 0s (0 seconds).

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

The animation-iteration-count property specifies the number of times an animation should run.

The animation-direction property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

* normal - The animation is played as normal (forwards). This is default
* reverse - The animation is played in reverse direction (backwards)
* alternate - The animation is played forwards first, then backwards
* alternate-reverse - The animation is played backwards first, then forwards

The z-index property specifies the stack order of an element.

An element with greater stack order is always in front of an element with a lower stack order.

**Note:** z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of [display:flex](https://www.w3schools.com/csSref/pr_class_display.php) elements).

**Note:** If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

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.

The following example shows the different behavior of display: inline, display: inline-block and display: block:

### **Example**

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;  
}

[Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss_inline-block_span1)

## Using inline-block to Create Navigation Links

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

The box-shadow property attaches one or more shadows to an element.

The scroll-behavior property specifies whether to smoothly animate the scroll position, instead of a straight jump, when the user clicks on a link within a scrollable box.

## Property Values

|  |  |
| --- | --- |
| **Value** | **Description** |
| auto | Allows a straight jump "scroll effect" between elements within the scrolling box. This is default |
| smooth | Allows a smooth animated "scroll effect" between elements within the scrolling box. |
| initial | Sets this property to its default value. [Read about initial](https://www.w3schools.com/cssref/css_initial.php) |
| inherit | Inherits this property from its parent element. [Read about inherit](https://www.w3schools.com/cssref/css_inherit.php) |

## How To Create Custom Scrollbars

Chrome, Edge, Safari and Opera support the non-standard ::-webkit-scrollbar pseudo element, which allows us to modify the look of the browser's scrollbar.

For webkit browsers, you can use the following pseudo elements to customize the browser's scrollbar:

* ::-webkit-scrollbar the scrollbar.
* ::-webkit-scrollbar-button the buttons on the scrollbar (arrows pointing upwards and downwards).
* ::-webkit-scrollbar-thumb the draggable scrolling handle.
* ::-webkit-scrollbar-track the track (progress bar) of the scrollbar.
* ::-webkit-scrollbar-track-piece the track (progress bar) NOT covered by the handle.
* ::-webkit-scrollbar-corner the bottom corner of the scrollbar, where both horizontal and vertical scrollbars meet.
* ::-webkit-resizer the draggable resizing handle that appears at the bottom corner of some elements.

The transition property is a shorthand property for:

* [transition-property](https://www.w3schools.com/cssref/css3_pr_transition-property.php)
* [transition-duration](https://www.w3schools.com/cssref/css3_pr_transition-duration.php)
* [transition-timing-function](https://www.w3schools.com/cssref/css3_pr_transition-timing-function.php)
* [transition-delay](https://www.w3schools.com/cssref/css3_pr_transition-delay.php)

**Note:** Always specify the [transition-duration](https://www.w3schools.com/cssref/css3_pr_transition-duration.php) property, otherwise the duration is 0s, and the transition will have no effect.

## Property Values

|  |  |
| --- | --- |
| **Value** | **Description** |
| [*transition-property*](https://www.w3schools.com/cssref/css3_pr_transition-property.php) | Specifies the name of the CSS property the transition effect is for |
| [*transition-duration*](https://www.w3schools.com/cssref/css3_pr_transition-duration.php) | Specifies how many seconds or milliseconds the transition effect takes to complete |
| [*transition-timing-function*](https://www.w3schools.com/cssref/css3_pr_transition-timing-function.php) | Specifies the speed curve of the transition effect |
| [*transition-delay*](https://www.w3schools.com/cssref/css3_pr_transition-delay.php) | Defines when the transition effect will start |
| initial | Sets this property to its default value. [Read about initial](https://www.w3schools.com/cssref/css_initial.php) |
| inherit | Inherits this property from its parent element. [Read about inherit](https://www.w3schools.com/cssref/css_inherit.php) |

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

Mouse over the element below to see a 2D transformation:

**2D rotate**

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:

### **Example**

* div {  
    transform: translate(50px, 100px);  
  }
* [Try it Yourself »](https://www.w3schools.com/css/tryit.asp?filename=trycss3_transform_translate)

## The rotate() Method



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

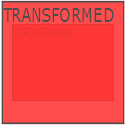
The following example rotates the <div> element clockwise with 20 degrees:

### **Example**

div {  
  transform: rotate(20deg);  
}

Using negative values will rotate the element counter-clockwise.

## The scale() Method



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

The following example increases the <div> element to be two times of its original width, and three times of its original height:

### **Example**

div {  
  transform: scale(2, 3);  
}

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

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

## RGBA Colors

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 (fully opaque).