The default value is linear.

Here we set ease-in.

.box {

    width: 100px;

    height: 100px;

    background-color: lightgreen;

    padding: 10px;

    transition: transform 0.5s ease-in;

    -webkit-transition: transform 0.5s ease-in;

    -moz-transition: transform 0.5s ease-in;

    -ms-transition: transform 0.5s ease-in;

    -o-transition: transform 0.5s ease-in;

}

.box:hover {

    transform: rotate(45deg);

    -webkit-transform: rotate(45deg);

    -moz-transform: rotate(45deg);

    -ms-transform: rotate(45deg);

    -o-transform: rotate(45deg);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The fourth parameter is the delay function:

transition: transform 0.5s cubic-bezier(.52,.91,1,.48) 1s;

-webkit-transition: transform 0.5s cubic-bezier(.52,.91,1,.48) 1s;

-moz-transition: transform 0.5s cubic-bezier(.52,.91,1,.48) 1s;

-ms-transition: transform 0.5s cubic-bezier(.52,.91,1,.48) 1s;

-o-transition: transform 0.5s cubic-bezier(.52,.91,1,.48) 1s;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@keyframes jafar-loka {

    0% {

        transform: scale(1);

        -webkit-transform: scale(1);

        -moz-transform: scale(1);

        -ms-transform: scale(1);

        -o-transform: scale(1);

    }

    25% {

        transform: scale(1.5);

        -webkit-transform: scale(1.5);

        -moz-transform: scale(1.5);

        -ms-transform: scale(1.5);

        -o-transform: scale(1.5);

    }

    50% {

        transform: rotate(45deg);

        -webkit-transform: rotate(45deg);

        -moz-transform: rotate(45deg);

        -ms-transform: rotate(45deg);

        -o-transform: rotate(45deg);

        background-color: lightgrey;

    }

    75% {

        transform: scale(1.2);

        -webkit-transform: scale(1.2);

        -moz-transform: scale(1.2);

        -ms-transform: scale(1.2);

        -o-transform: scale(1.2);

    }

    100% {

        transform: rotate(0) scale(1);

        -webkit-transform: rotate(0) scale(1);

        -moz-transform: rotate(0) scale(1);

        -ms-transform: rotate(0) scale(1);

        -o-transform: rotate(0) scale(1);

    }

}

.box {

    width: 100px;

    height: 100px;

    padding: 10px;

    background-color: lightgreen;

    animation-name: jafar-loka;

    animation-duration: 4s;

    animation-delay: 0.5s;

    animation-iteration-count: 4; /\* This specify the number of times to run the animation\*/

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Note:*** The animation-iteration-count can have infinite-value to run the animation in loop.

***Note:*** Also, we can define the timing function:

animation-timing-function: ease-out;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The alternate: The animation cycle iterations that are odd counts are played in the normal direction, and the animation cycle iterations that are even counts are played in a reverse direction.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***Note:*** Here also we have many values, like: reverse, normal(the default), …etc.

animation-direction: alternate;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Also, we have animation-fill-mode.

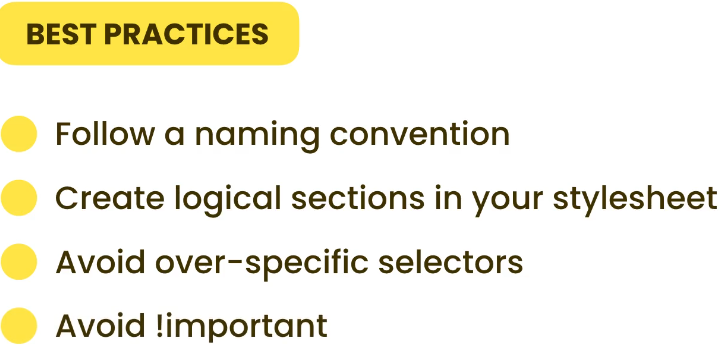
/\* animation: name duration timing-function delay iteration-count direction fill-mode; \*/

animation-fill-mode: backwards;

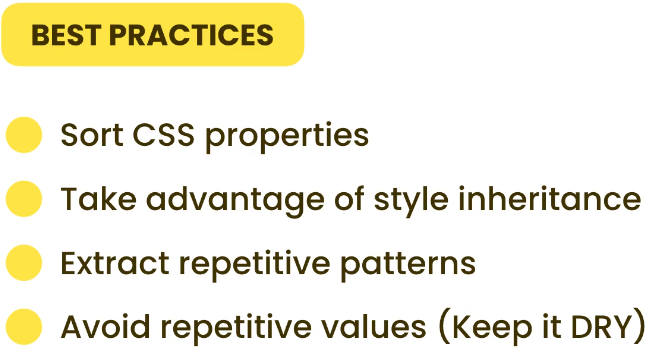
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

For name convention; if we set prefix for class-names, we don’t need to use nesting.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

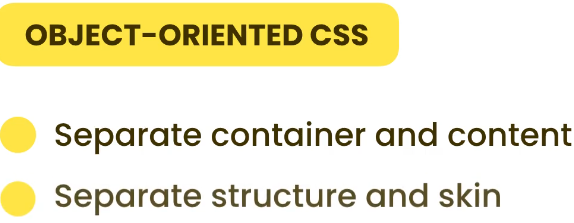
This will create two-divs:



<div class="one"></div>

<div class="two"></div>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

***BEM:*** for Naming convention in CSS styles:

Separate the name of classes between two \_: \_\_ (separate the block from element)

***Ex:*** card\_\_header.

Separate block from modifier; we use two -: --.

***Ex:*** card--popular.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Follow a naming convention for naming IDs and classes. The most common naming conventions are **PascalCase**, **camelCase** and **kabob-case**.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

For a small project, you can write all of your CSS rules in one stylesheet. Use CSS

comments to create logical sections in your stylesheet. For a more complex project, you

need to separate your stylesheet into multiple files and combine them together using

build tools like Webpack, Rollup or Parcel.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

When using a custom font, the user may experience a ***flash of unstyled text (FOUT)***. Some browsers display text using a fallback font while downloading the custom font and swap it once the custom font is available. This may cause a layout shift depending on how the content is structured. Some browsers hide the text initially while downloading the custom font. This causes a ***flash of invisible text (FOIT).*** Using the font-display property we can tell the browser how to handle this situation.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

It’s best to use the rem unit for vertical margins. For headings, the top margin should be

noticeably greater than the bottom margin so the heading gets separated from the text

before and gets connected to the text after.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Using the ***line-height property*** we can specify the height of lines. It’s best to set this

property to a unitless value ***around 1.5***. This value will be **multiplied** by ***the font size*** of

the current element so we don’t need to remember to ***change the line height*** if we

modify the ***font size***.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The three properties used for horizontal spacing are: letter-spacing, word-spacing, and width. It’s often better to apply a negative letter spacing to headings so

they look more compact.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The ideal line length is about 60-70 characters. We can achieve that by applying a width

of 50ch. The ch unit represents the width of the 0. 50 zeroes roughly represents 60-70

characters because some characters (like i and 1) are narrower than 0.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To ensure that the container is in the middle, set the horizontal margin to **auto**. This

way any extra space will be equally distributed to the left and right margins.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Give the image a width of 100% to make it responsive.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Margin collapsing happens when the top and bottom margins of elements are combined into a single margin. The size of the margin is equal to the largest of the two margins.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Using the position property we can precisely position an element. The default value of this property is static. If we change the value of this property, the element is considered *positioned.*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

By setting the position to relative, we can position an element relative to its normal position. By setting it to absolute, we can position it relative to its positioned parent. That means, the parent (or ancestor) should be a positioned element. By setting the position to fixed, we can position the element relative to the viewport.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

We use the ***img element*** to display content images. Content images can represent

meaningful content or be used for decorative purposes. If used ***for decoration***, we

should set the alt attribute to ***an empty string***; otherwise, screen readers will read out

the name of the file which may be distracting to the user.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Note that the input fields and the button have the same width. One way to achieve this is by applying the same width (eg 200px) to both these elements. But what if we need

shorter or longer input fields on a different page? A more flexible approach is to give

these elements a width of 100% so they always stretch and fill their container. The actual width can then be applied on the container.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.menu{

    background: #2a2138;

    color: #cfc6dc;

}

.menu a {

    /\* I've used inherit here so if we change the color of the text in the

  .menu rule (above), all anchors in the menu will automatically inherit

  in the new color. Alternatively, I'd have to specify the same color in

  two places. \*/

    color: inherit;

    text-decoration: none;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

.form-control, .btn {

    border-radius: 5px;

    /\* The default box-sizing for input fields is content-box. So, any padding we apply to an input

    field, increases its width. To ensure that input fields and buttons have the same width, we need

    to set box-sizing to border-box. \*/

    box-sizing: border-box;

    padding: 1.5rem;

    /\* It's a common technique to set the width of input fields and buttons to 100% so they fill up

    their container. We can then control their size through their container (in this case form-signin) \*/

    width: 100%;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

In CSS, ***vmin*** stands for viewport minimum. The ***vmin*** ***function*** is used to set the size of an element as a percentage of the ***minimum*** value ***between*** ***the*** ***viewport*** ***width*** or ***height***. For example, if the ***viewport is 1000px wide*** and ***800px high***, if we set the width of an element to ***30vmin*** , it will ***be 30% of the height***.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*