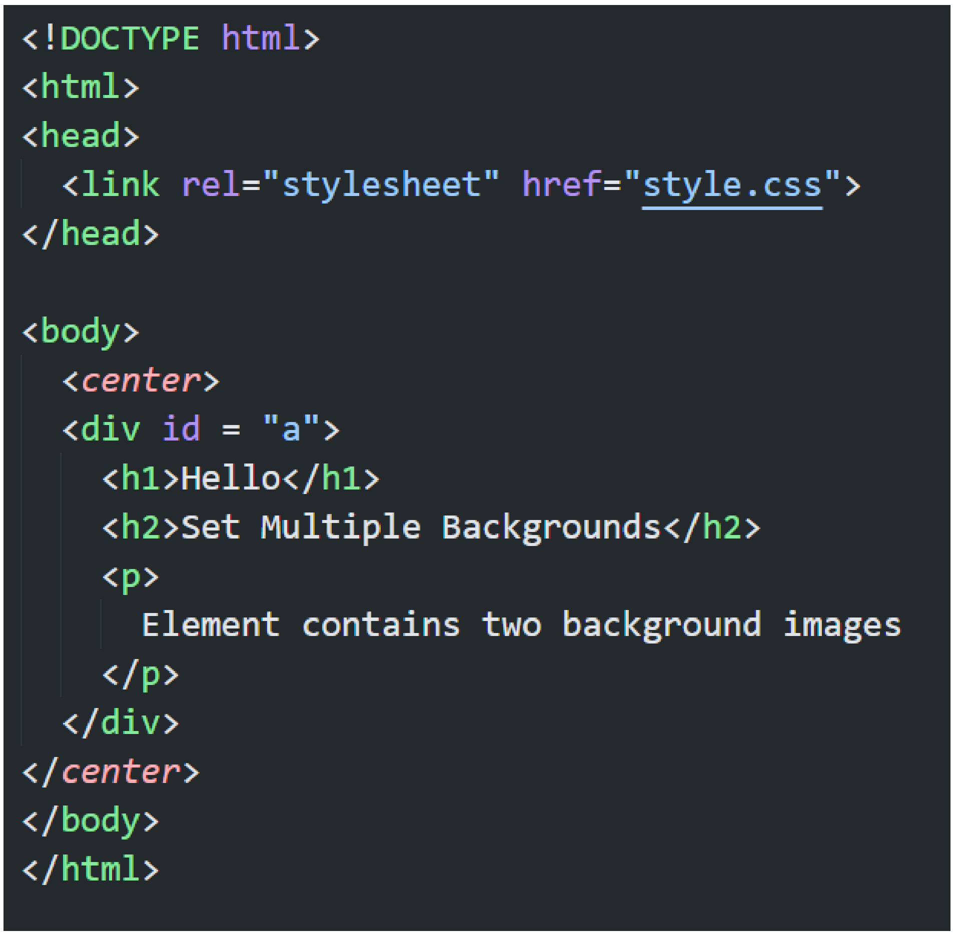
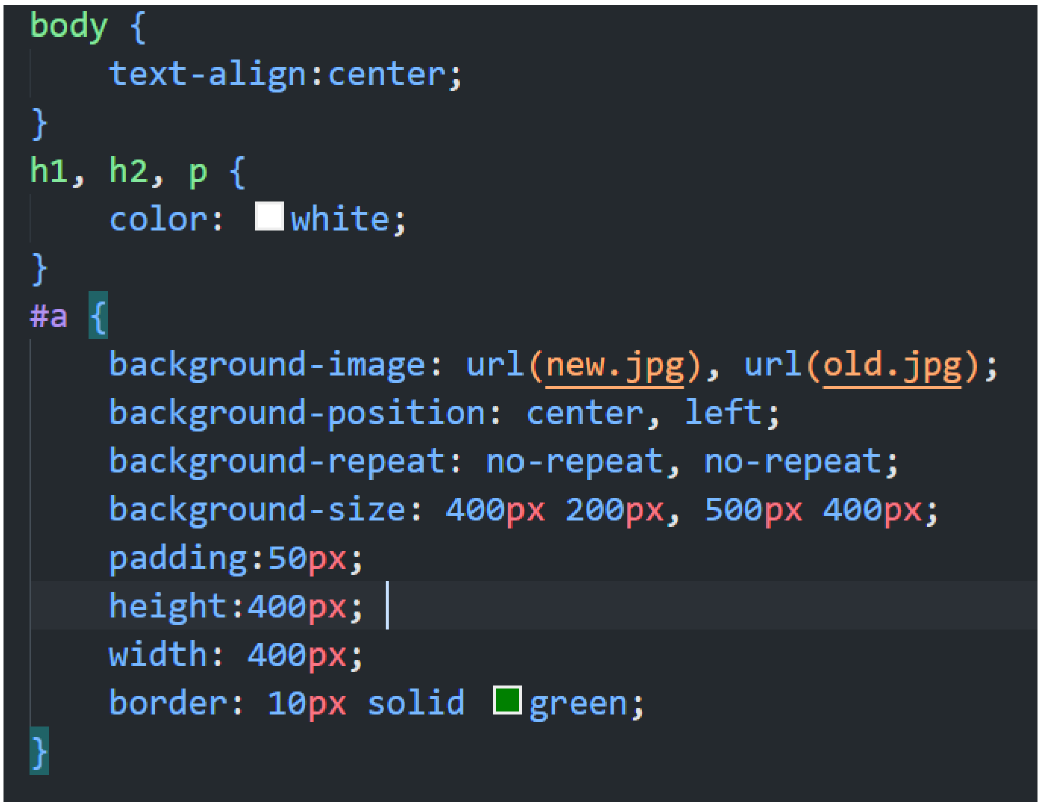
Back ground, multiple background=>

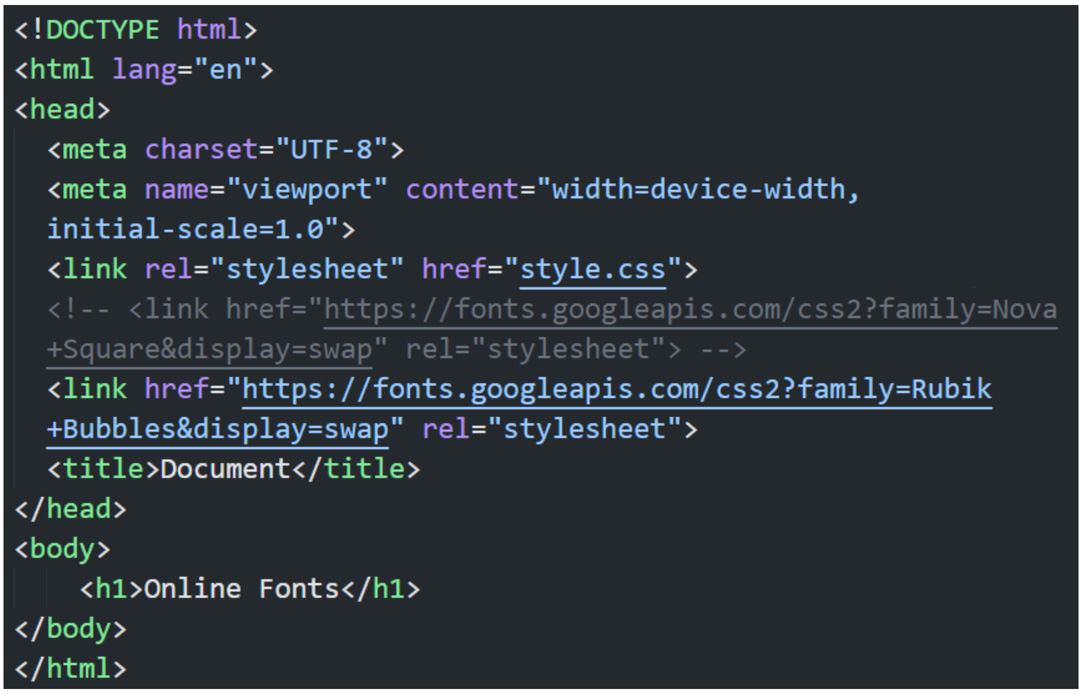


```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="style.css">
</head>

<body>
  <center>
  <div id = "a">
    <h1>Hello</h1>
    <h2>Set Multiple Backgrounds</h2>
    <p>
      Element contains two background images
    </p>
  </div>
</center>
</body>
</html>
```



```css
body {
    text-align:center;
}
h1, h2, p {
    color: white;
}
#a {
    background-image: url(new.jpg), url(old.jpg);
    background-position: center, left;
    background-repeat: no-repeat, no-repeat;
    background-size: 400px 200px, 500px 400px;
    padding:50px;
    height:400px;
    width: 400px;
    border: 10px solid green;
}
```

Online Fonts=>



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <!-- <link href="https://fonts.googleapis.com/css2?family=Nova
  +Square&display=swap" rel="stylesheet"> -->
  <link href="https://fonts.googleapis.com/css2?family=Rubik
  +Bubbles&display=swap" rel="stylesheet">
  <title>Document</title>
</head>
<body>
    <h1>Online Fonts</h1>
</body>
</html>
```

Download fonts=>

https://www.dafont.com/ => download font => extract that font =>

install that font

@font-face {

  font-family: mine;   //this s the custom name that we want to give to our font

  src: url(da.otf);   //this is the font name with extention. The font should be in the fame folder where we are coding.

}

body {

  margin: 0;

  font-family: mine;   //here we are calling that name in using that font

  background-color: #212121;

}

Text Effects =>

text-overflow

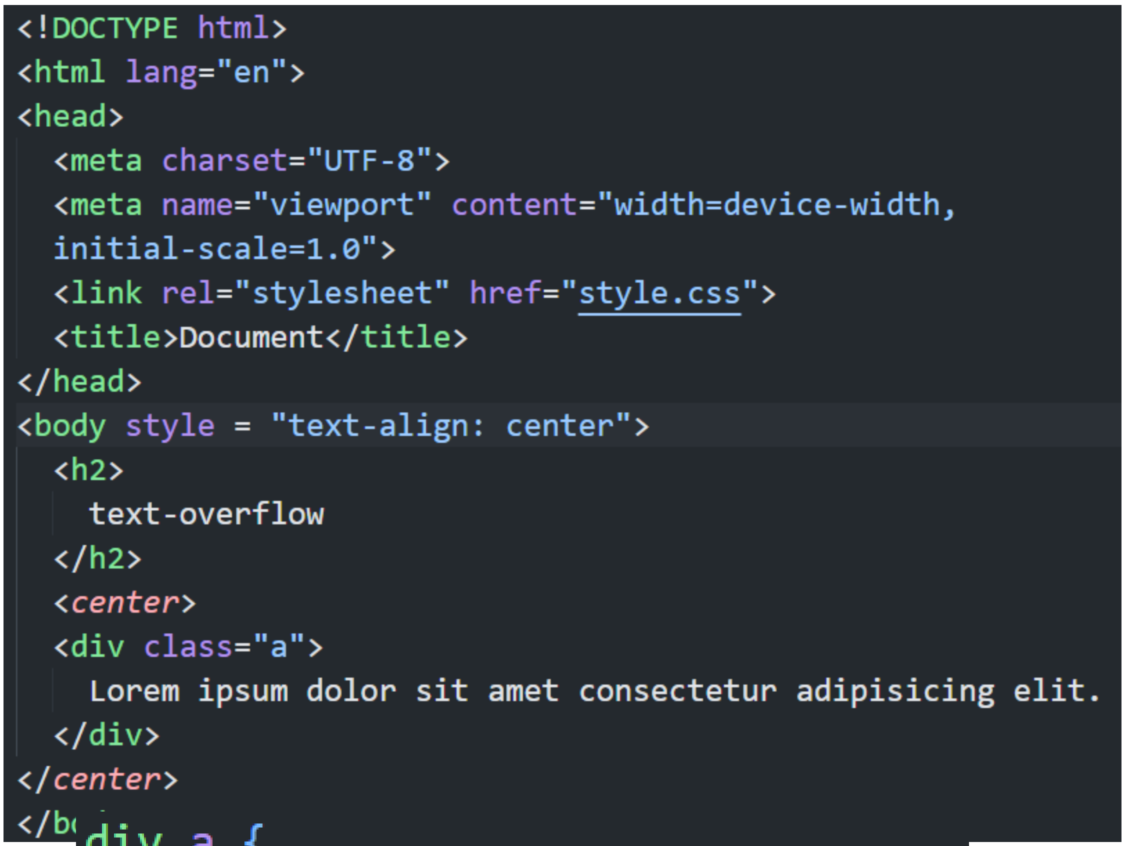
word-wrap

word-break

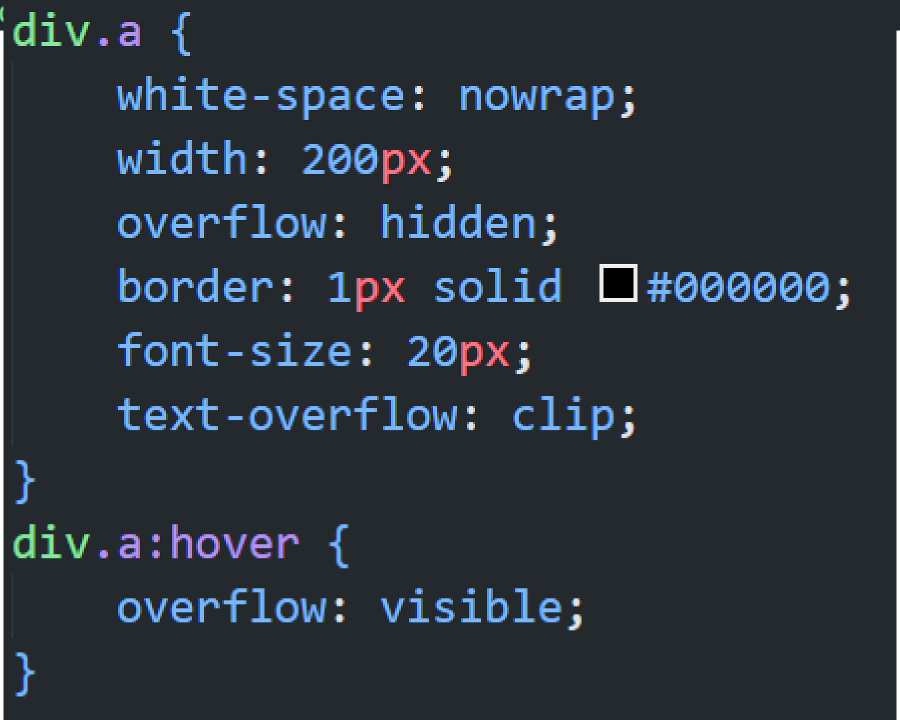
writing-mode

Text-Overflow: The CSS Text overflow property is a way to limit text that exceeds the width of it’s parent. It helps to specify the way to represent the portion of overflowing text which is not visible to the user.

```
element {
    text-overflow: clip | ellipsis;
    //CSS Property
}
```

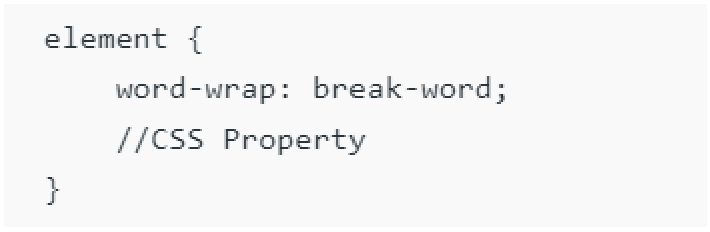


```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Document</title>
</head>
<body style = "text-align: center">
  <h2>
    text-overflow
  </h2>
  <center>
  <div class="a">
    Lorem ipsum dolor sit amet consectetur adipisicing elit.
  </div>
</center>
</b
```

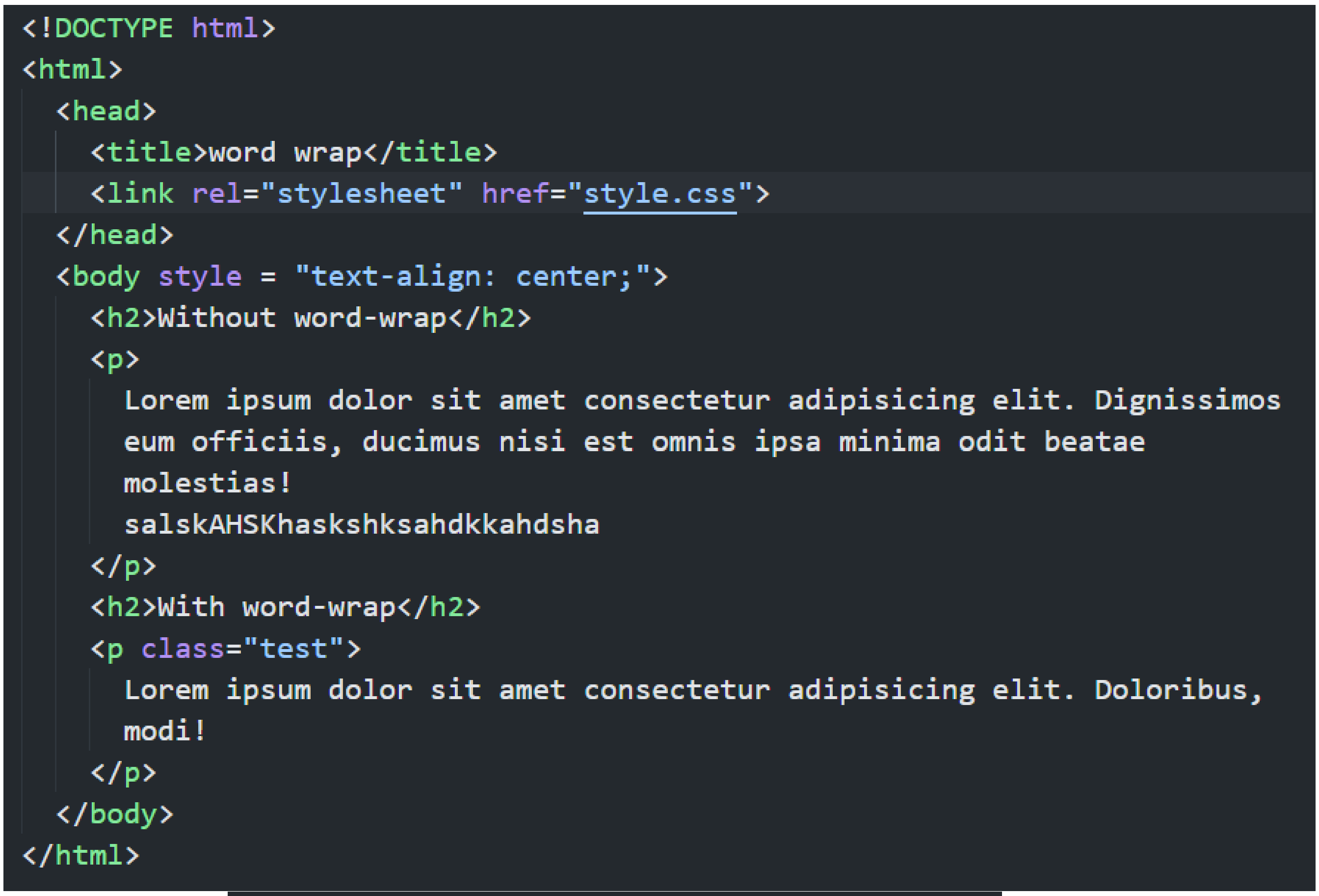


```css
div.a {
    white-space: nowrap;
    width: 200px;
    overflow: hidden;
    border: 1px solid #000000;
    font-size: 20px;
    text-overflow: clip;
}
div.a:hover {
    overflow: visible;
}
```

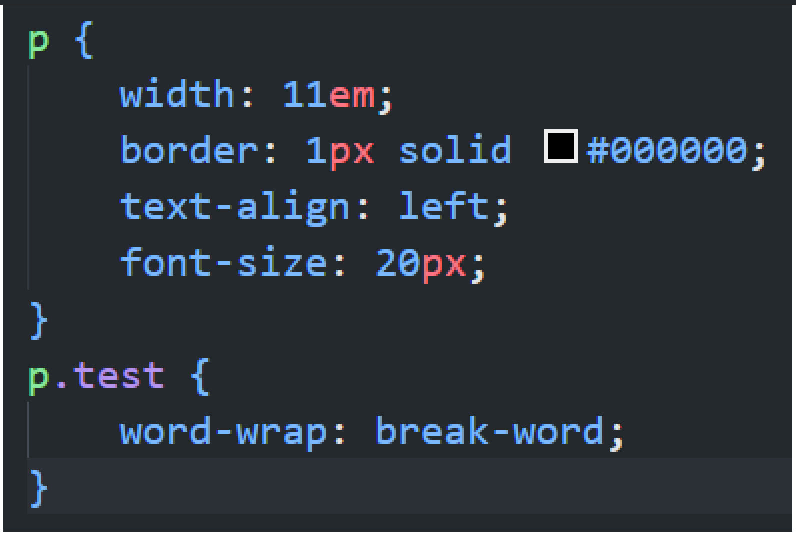
Word wrap: The CSS word-wrap property defines whether the browser is allowed to line break within words when a word is too long to fit within its parent container. If a word is too long to fit within an area, it expands outside:



```
element {
    word-wrap: break-word;
    //CSS Property
}
```



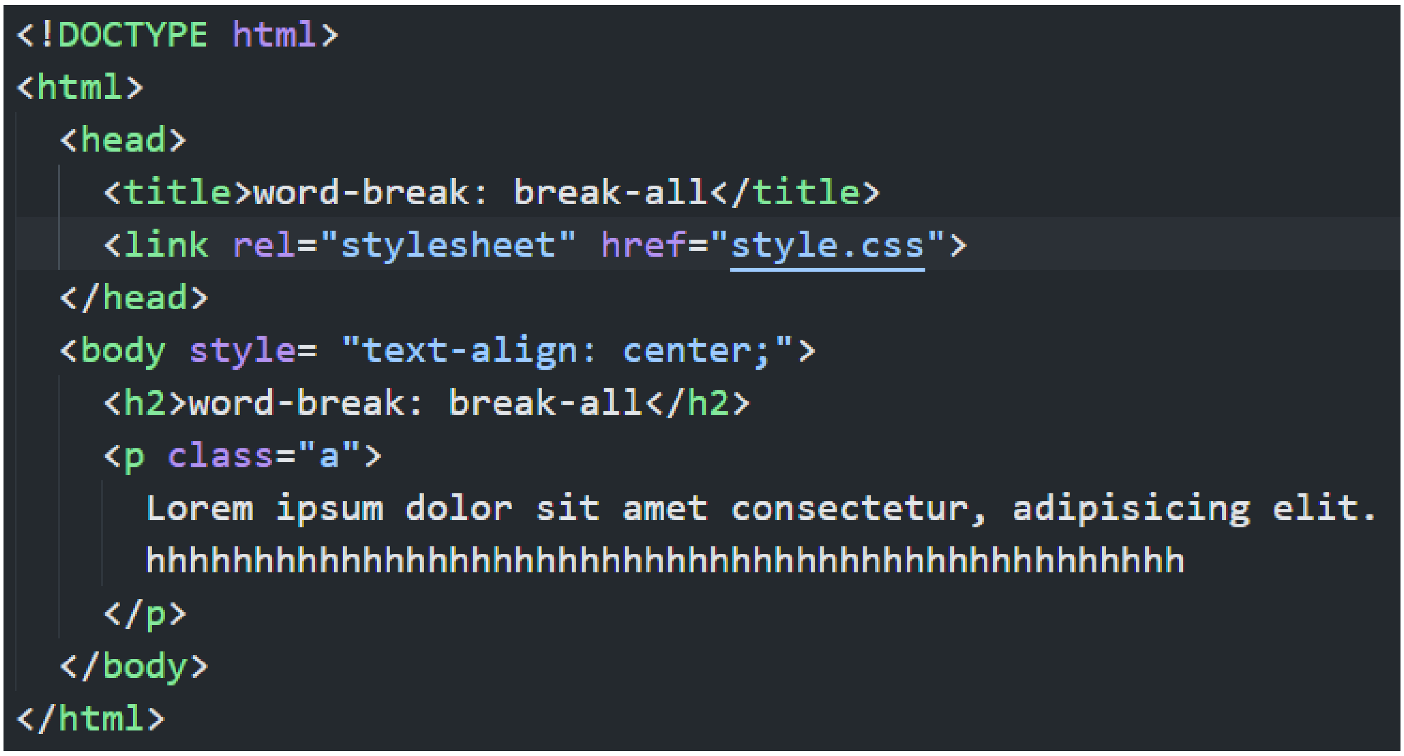
```html
<!DOCTYPE html>
<html>
  <head>
    <title>word wrap</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body style = "text-align: center;">
    <h2>Without word-wrap</h2>
    <p>
      Lorem ipsum dolor sit amet consectetur adipisicing elit. Dignissimos
      eum officiis, ducimus nisi est omnis ipsa minima odit beatae
      molestias!
      salskAHSKhaskshksahdkkahdsha
    </p>
    <h2>With word-wrap</h2>
    <p class="test">
      Lorem ipsum dolor sit amet consectetur adipisicing elit. Doloribus,
      modi!
    </p>
  </body>
</html>
```



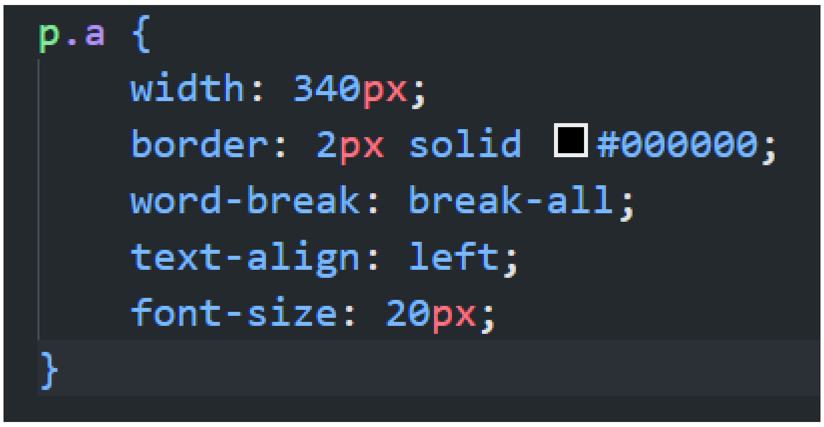
```css
p {
    width: 11em;
    border: 1px solid ■#000000;
    text-align: left;
    font-size: 20px;
}
p.test {
    word-wrap: break-word;
}
```

Word breaking: The word-break CSS property sets whether line breaks appear wherever the text would otherwise overflow its content box. It specifies line breaking rules.

```
element {
     word-break: keep-all | break-all;
     //CSS Property
}
```

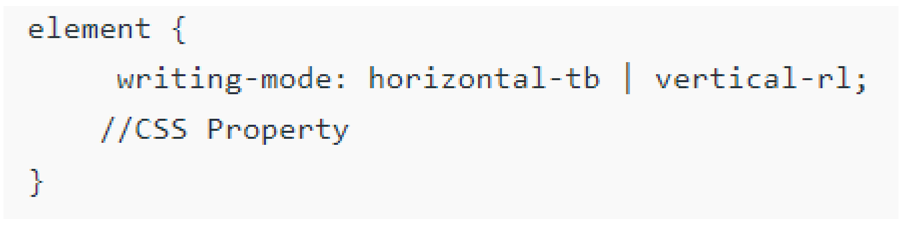


```html
<!DOCTYPE html>
<html>
  <head>
    <title>word-break: break-all</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body style= "text-align: center;">
    <h2>word-break: break-all</h2>
    <p class="a">
      Lorem ipsum dolor sit amet consectetur, adipisicing elit.
      hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
    </p>
  </body>
</html>
```

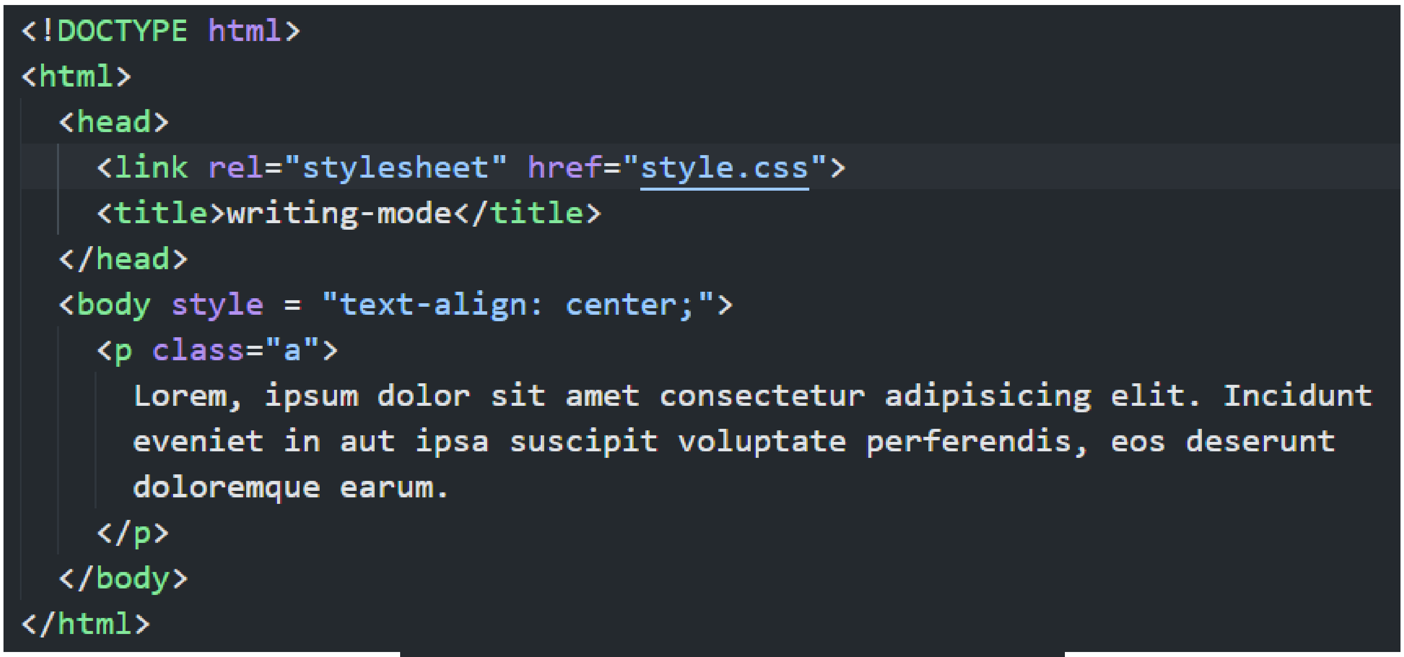


```css
p.a {
    width: 340px;
    border: 2px solid #000000;
    word-break: break-all;
    text-align: left;
    font-size: 20px;
}
```

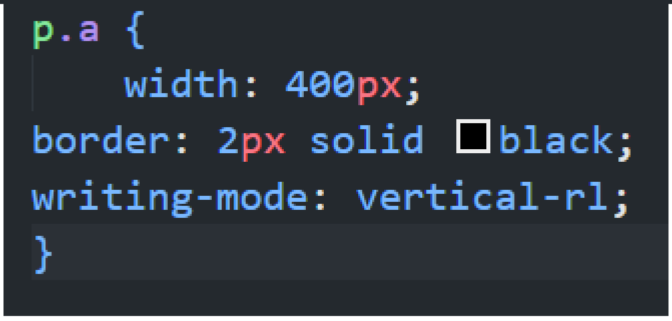
Writing mode: The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically. tb(top to bottom) default, rl (right to left)



```
element {
      writing-mode: horizontal-tb | vertical-rl;
    //CSS Property
}
```

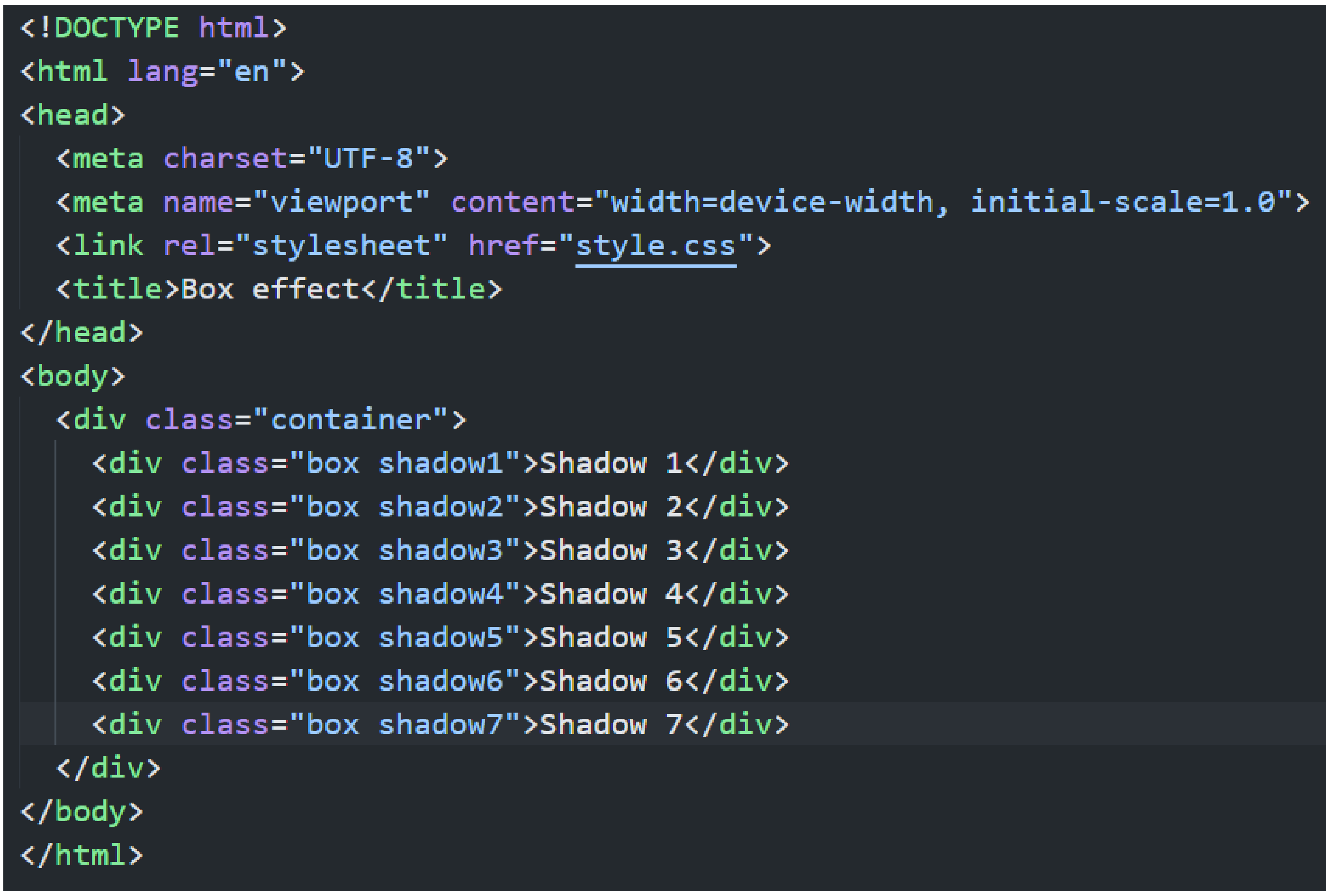


```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="style.css">
    <title>writing-mode</title>
  </head>
  <body style = "text-align: center;">
    <p class="a">
      Lorem, ipsum dolor sit amet consectetur adipisicing elit. Incidunt
      eveniet in aut ipsa suscipit voluptate perferendis, eos deserunt
      doloremque earum.
    </p>
  </body>
</html>
```

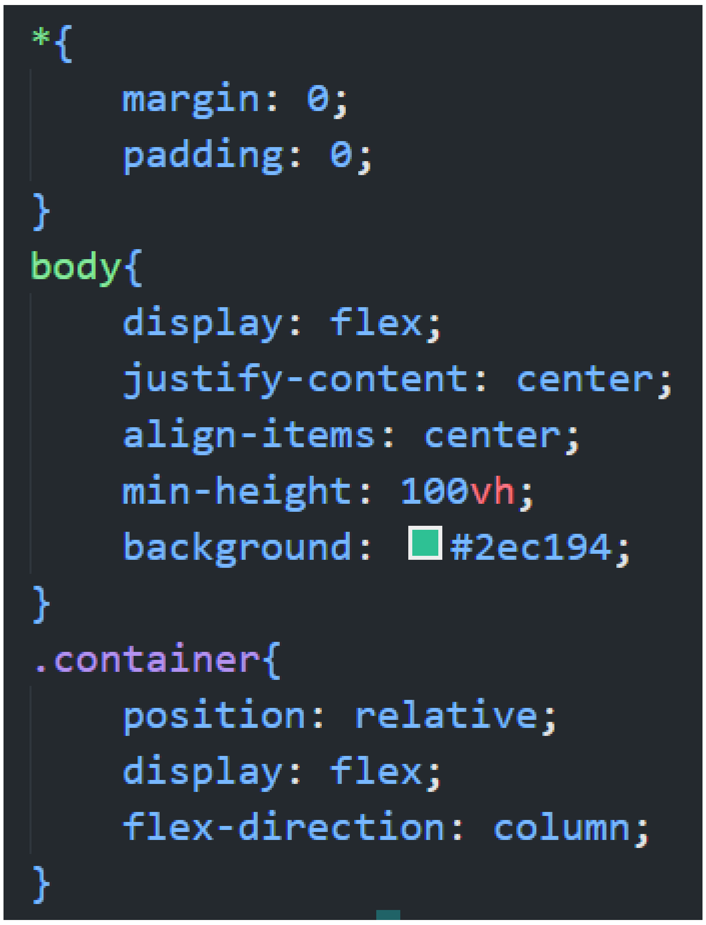


```
p.a {
    width: 400px;
border: 2px solid black;
writing-mode: vertical-rl;
}
```

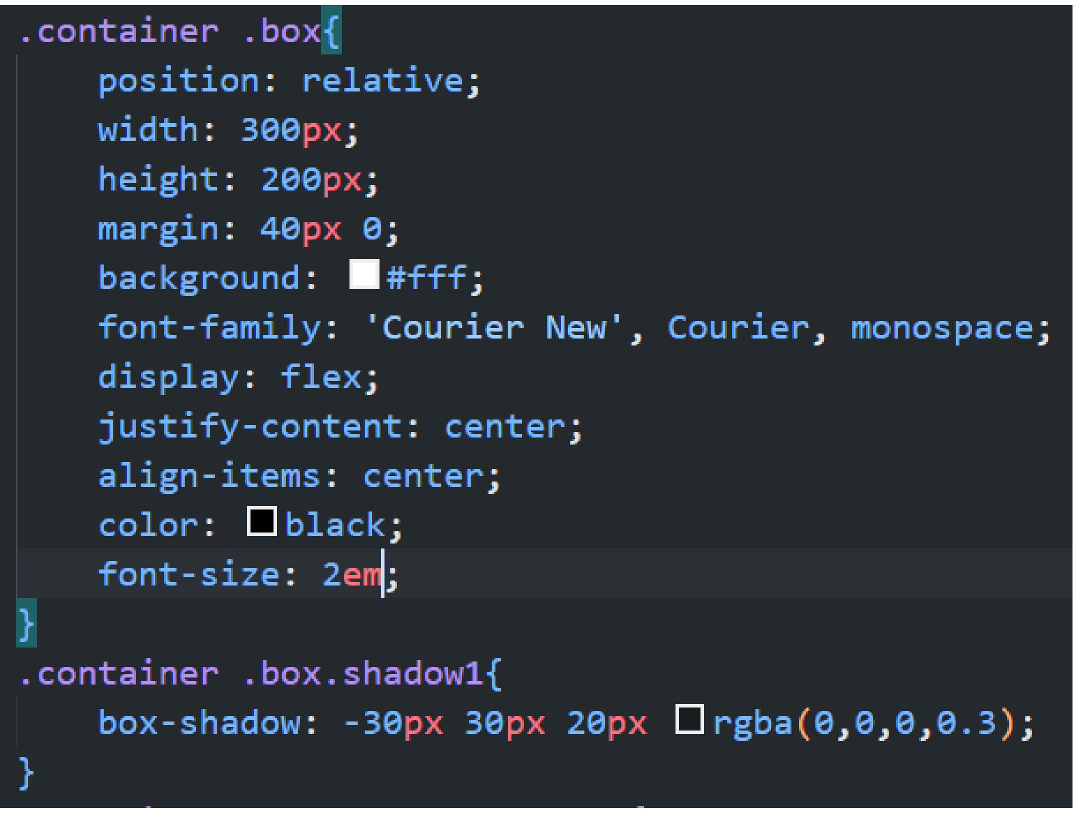
Box effect =>



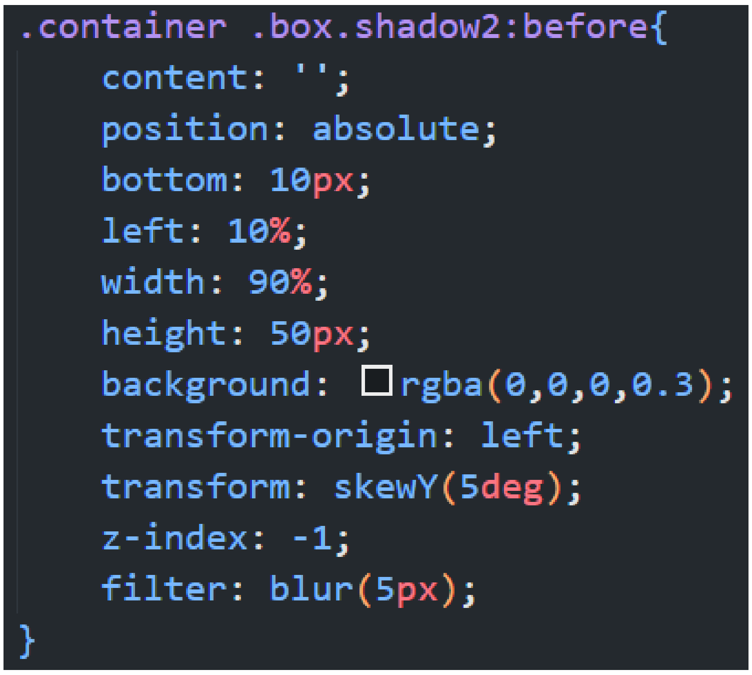
```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Box effect</title>
</head>
<body>
  <div class="container">
    <div class="box shadow1">Shadow 1</div>
    <div class="box shadow2">Shadow 2</div>
    <div class="box shadow3">Shadow 3</div>
    <div class="box shadow4">Shadow 4</div>
    <div class="box shadow5">Shadow 5</div>
    <div class="box shadow6">Shadow 6</div>
    <div class="box shadow7">Shadow 7</div>
  </div>
</body>
</html>
```



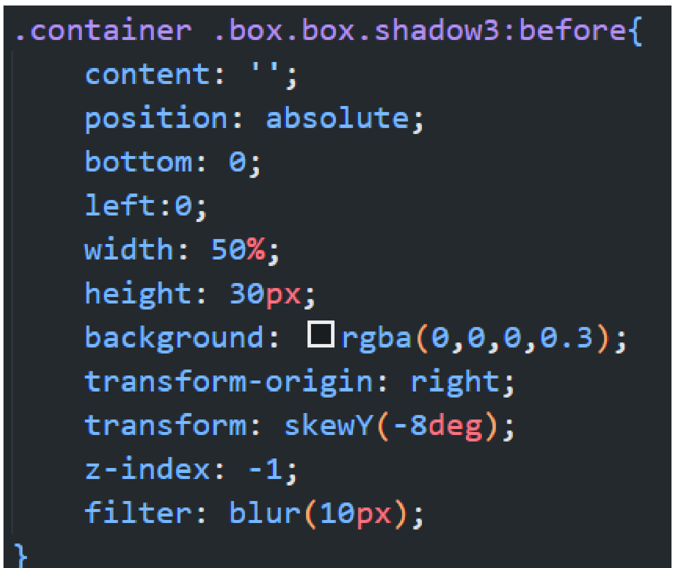
```css
*{
    margin: 0;
    padding: 0;
}
body{
    display: flex;
    justify-content: center;
    align-items: center;
    min-height: 100vh;
    background: #2ec194;
}
.container{
    position: relative;
    display: flex;
    flex-direction: column;
}
```



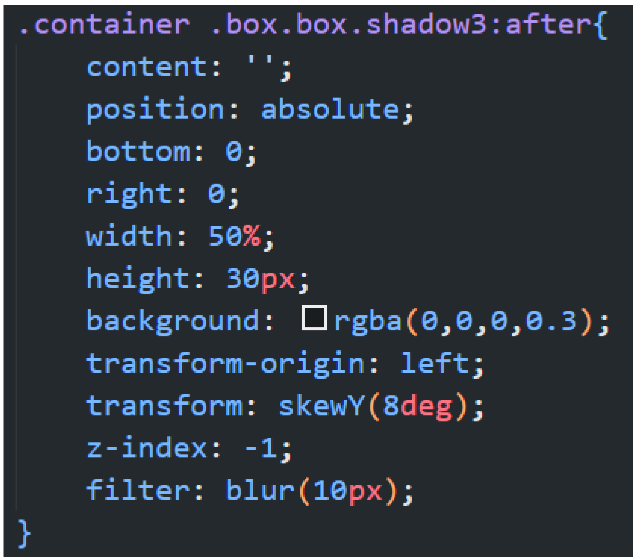
```css
.container .box{
    position: relative;
    width: 300px;
    height: 200px;
    margin: 40px 0;
    background: #fff;
    font-family: 'Courier New', Courier, monospace;
    display: flex;
    justify-content: center;
    align-items: center;
    color: black;
    font-size: 2em;
}
.container .box.shadow1{
    box-shadow: -30px 30px 20px rgba(0,0,0,0.3);
}
```



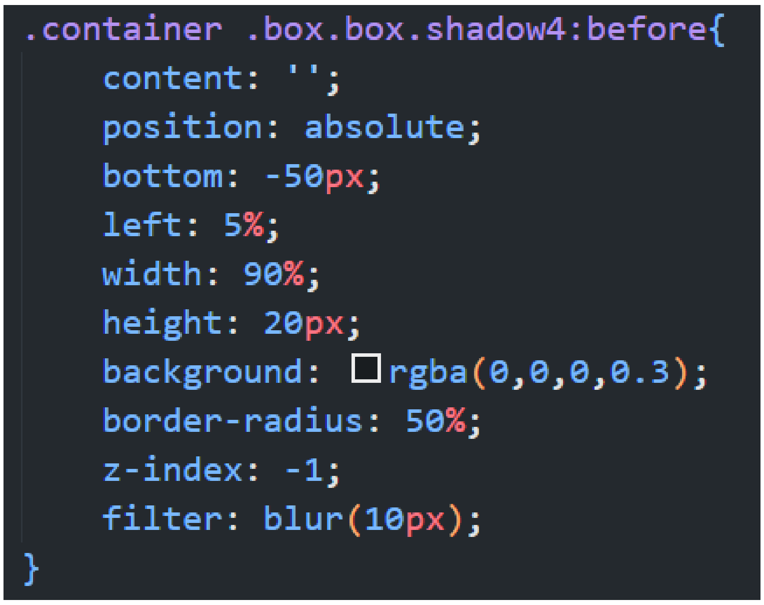
```css
.container .box.shadow2:before{
    content: '';
    position: absolute;
    bottom: 10px;
    left: 10%;
    width: 90%;
    height: 50px;
    background: rgba(0,0,0,0.3);
    transform-origin: left;
    transform: skewY(5deg);
    z-index: -1;
    filter: blur(5px);
}
```



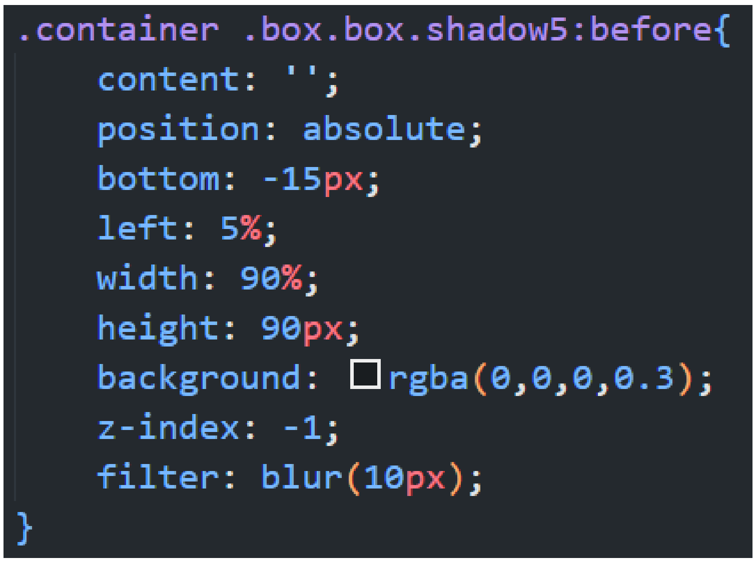
```css
.container .box.box.shadow3:before{
    content: '';
    position: absolute;
    bottom: 0;
    left:0;
    width: 50%;
    height: 30px;
    background: rgba(0,0,0,0.3);
    transform-origin: right;
    transform: skewY(-8deg);
    z-index: -1;
    filter: blur(10px);
}
```



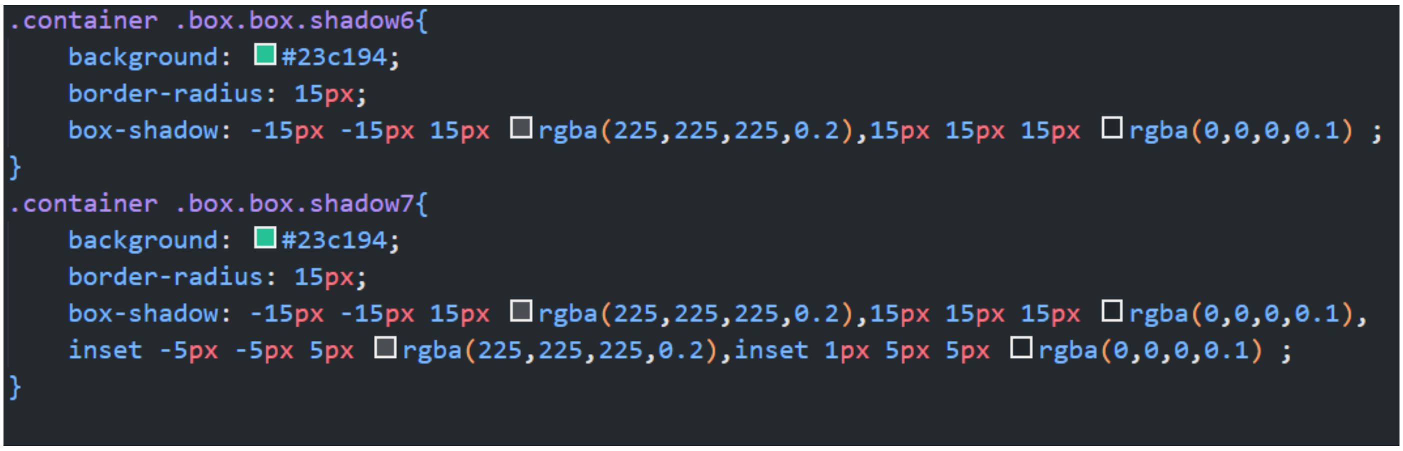
```css
.container .box.box.shadow3:after{
    content: '';
    position: absolute;
    bottom: 0;
    right: 0;
    width: 50%;
    height: 30px;
    background: rgba(0,0,0,0.3);
    transform-origin: left;
    transform: skewY(8deg);
    z-index: -1;
    filter: blur(10px);
}
```



```css
.container .box.box.shadow4:before{
    content: '';
    position: absolute;
    bottom: -50px;
    left: 5%;
    width: 90%;
    height: 20px;
    background: rgba(0,0,0,0.3);
    border-radius: 50%;
    z-index: -1;
    filter: blur(10px);
}
```



```css
.container .box.box.shadow5:before{
    content: '';
    position: absolute;
    bottom: -15px;
    left: 5%;
    width: 90%;
    height: 90px;
    background: rgba(0,0,0,0.3);
    z-index: -1;
    filter: blur(10px);
}
```

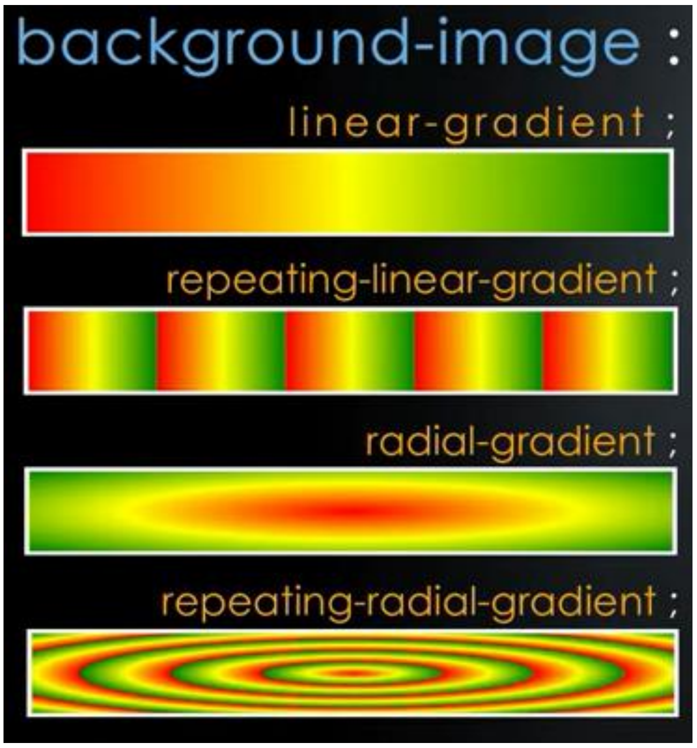


```css
.container .box.box.shadow6{
    background: #23c194;
    border-radius: 15px;
    box-shadow: -15px -15px 15px rgba(225,225,225,0.2),15px 15px 15px rgba(0,0,0,0.1) ;
}
.container .box.box.shadow7{
    background: #23c194;
    border-radius: 15px;
    box-shadow: -15px -15px 15px rgba(225,225,225,0.2),15px 15px 15px rgba(0,0,0,0.1),
    inset -5px -5px 5px rgba(225,225,225,0.2),inset 1px 5px 5px rgba(0,0,0,0.1) ;
}
```

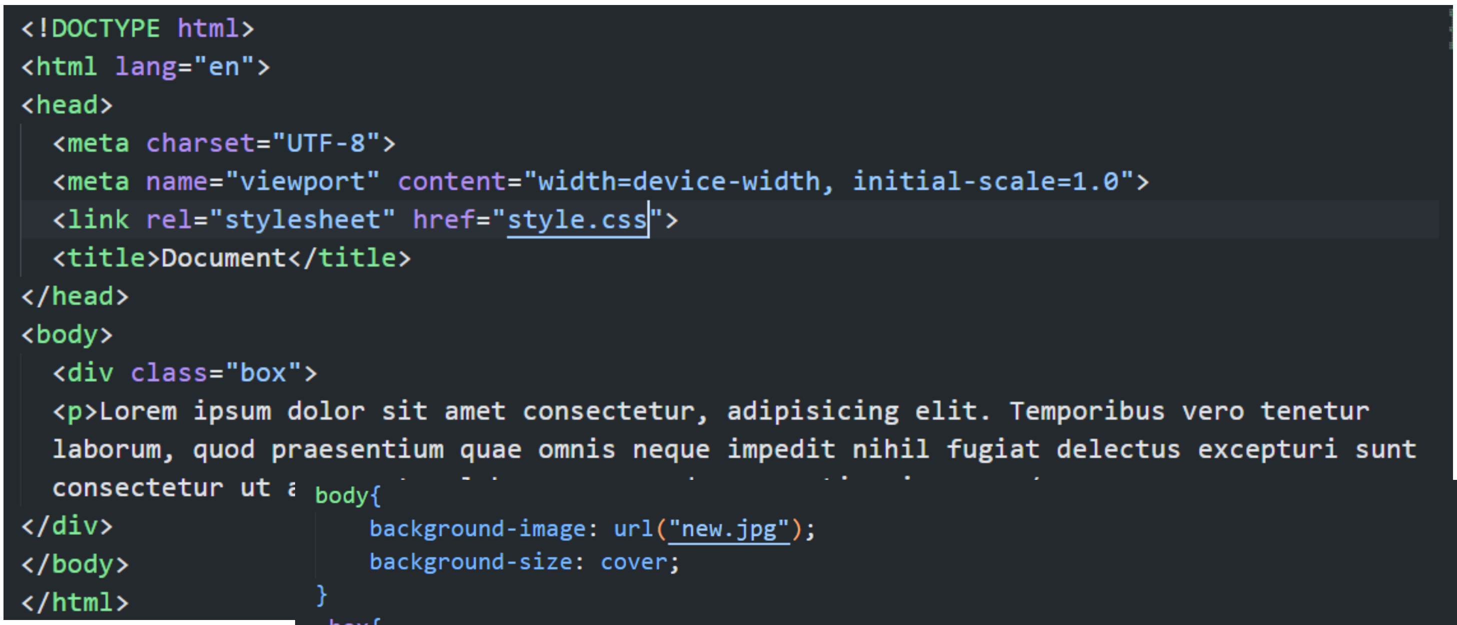
Gradient =>

By the help of Gradient we can transfer from one or more color to another color smoothly. In Css gradient is use with the background image property. Tyes of gradients linear gradient linear gradient, repeating linear gradient, radial gradient repeating radial gradient.

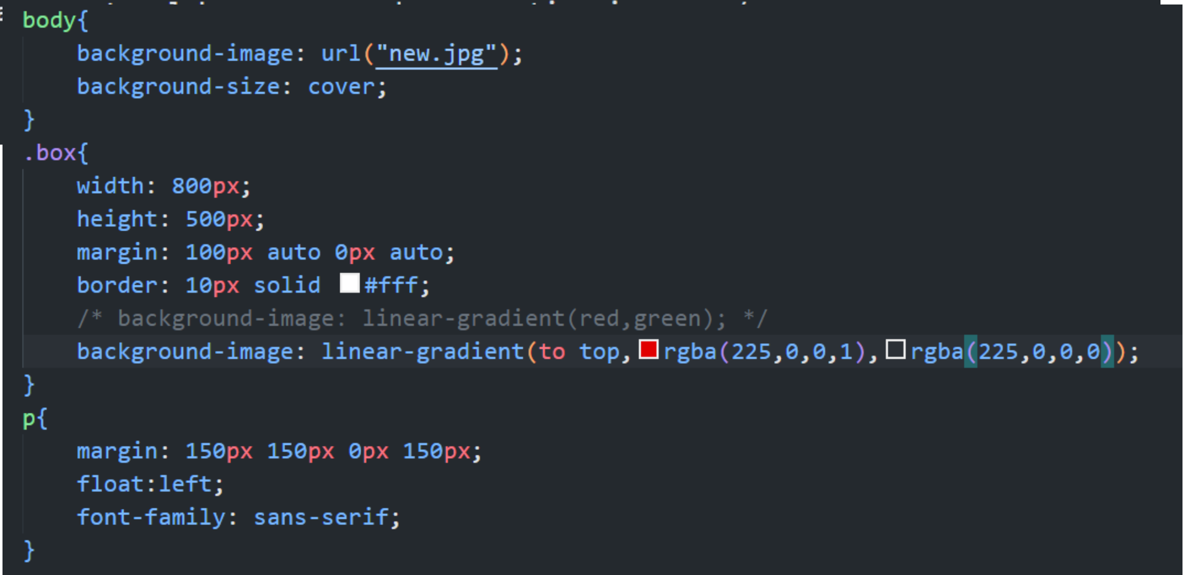
Linear gradient =>



Color in gradient moves form top to bottom this is its by default value.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="style.css">
  <title>Document</title>
</head>
<body>
  <div class="box">
  <p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Temporibus vero tenetur
  laborum, quod praesentium quae omnis neque impedit nihil fugiat delectus excepturi sunt
  consectetur ut a
</div>
</body>
</html>
```



```
body{
    background-image: url("new.jpg");
    background-size: cover;
}
.box{
    width: 800px;
    height: 500px;
    margin: 100px auto 0px auto;
    border: 10px solid #fff;
    /* background-image: linear-gradient(red,green); */
    background-image: linear-gradient(to top,rgba(225,0,0,1),rgba(225,0,0,0));
}
p{
    margin: 150px 150px 0px 150px;
    float:left;
    font-family: sans-serif;
}
```

To reverse this movement we have to use -> to top -> in front of that color.

For right to left -> to right.

For left to right -> to left.

From corner -> to bottom right,

to bottom left, to top left ,

to top right.

For the movement in angle-> 45deg, red, yellow,

-90deg, red, yellow.

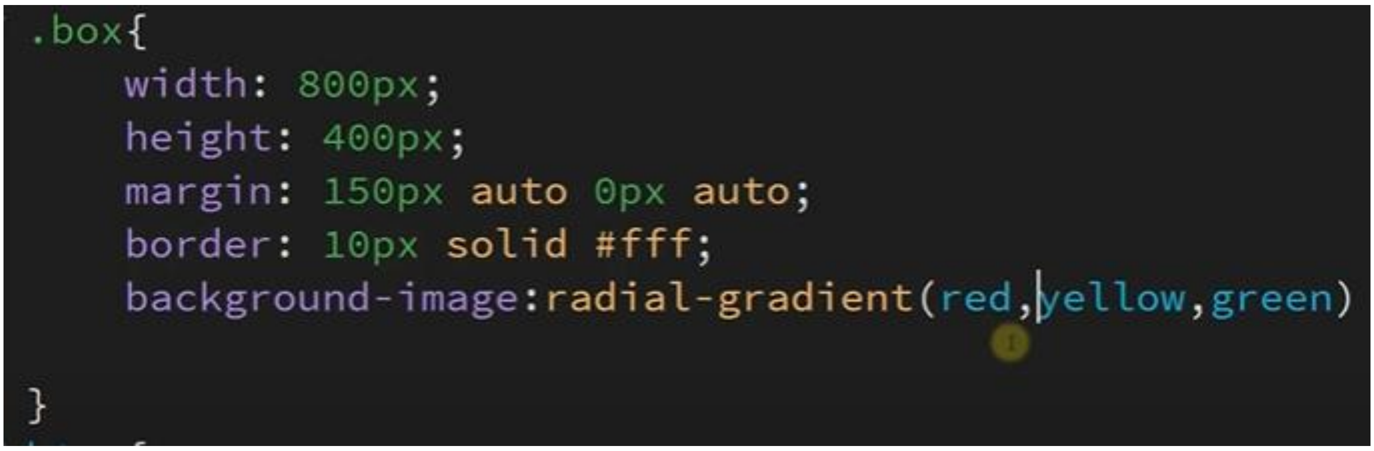
Colors in percentage -> to right red, yellow 50%. For multi colors -> to right, red, yellow, green, pink, blue.

For transparency in colors-> to right,rgba(225,0, 0, 1),

rgba(225,0,0,0) -> 0 means transparent, 1 means not transparent, .5 means 50% transaparent.

Radial gradient =>

The radial gradient define a center and according to this circle transition on color takes place. To perform radial gradient we must have two colors in it.



```
.box{
    width: 800px;
    height: 400px;
    margin: 150px auto 0px auto;
    border: 10px solid #fff;
    background-image:radial-gradient(red,yellow,green)

}
```

In this colors can be in percentage also -> red 30%, yellow 20%, green 40%.

In Radial gradient the shape of the color is by default that is elliptical shape we can change this shape according to us-> circle, red 30%, yellow 20%, green 40%.

We can also change the location of this shape-> circle at top , red 30%, yellow 20%, green 40%.

circle at bottom, red 30%, yellow 20%, green 40%.

For corner-> circle at top right, red 30%, yellow 20%, green 40%.

For corner-> circle at top left, red 30%, yellow 20%, green 40%.

By using percentange =>circle at 0%, red 30%, yellow 20%, green 40%.

circle at 100% , red 30%, yellow 20%, green 40%.

Circle at 25%, red 30%, yellow 20%, green 40%.

First percentage is x-axis and other percentage is y-axis-> circle 25% 25%, red 30%, yellow 20%, green 40%.

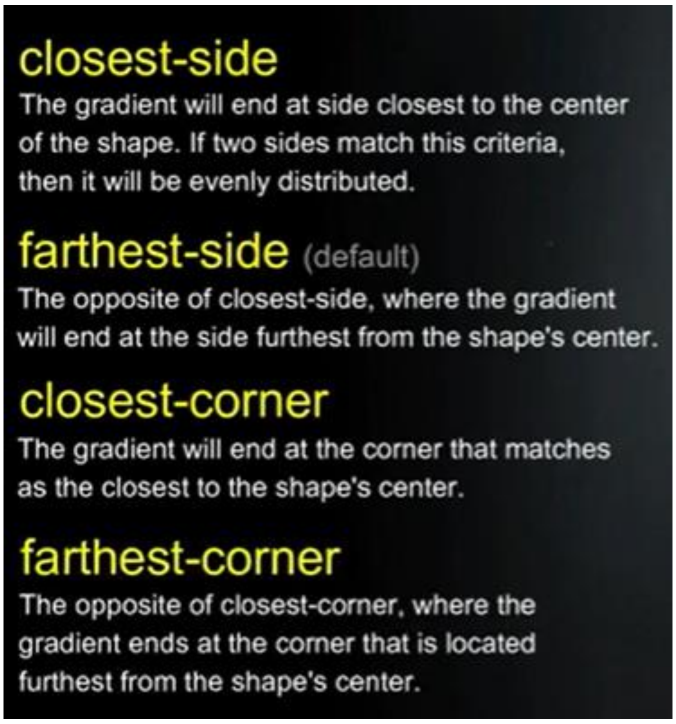
circle 25% 60%, red 30%, yellow 20%, green 40%.

circle 90% 90%, red 30%, yellow 20%, green 40%.

We can do transparency by rgba or we can do transparency directly-> circle at top , red, transparent.

We can also use two or more radial gradients. radial-gradient(at top red, transparent), radial-gradient(at top blue, transparent).

In redial gradient we can define the gradient side by the help of => radial-gradient(closest-side at 30% 40%, red,yellow,blue);



closest-side
The gradient will end at side closest to the center of the shape. If two sides match this criteria, then it will be evenly distributed.

farthest-side (default)
The opposite of closest-side, where the gradient will end at the side furthest from the shape's center.

closest-corner
The gradient will end at the corner that matches as the closest to the shape's center.

farthest-corner
The opposite of closest-corner, where the gradient ends at the corner that is located furthest from the shape's center.

Transforms in CSS =>

We can apply transform on 2d and 3d. The transform property in CSS is used to change the coordinate space of the visual formatting model. This is used to add effects like skew, rotate, translate, etc on elements. Transform means changing any component position or size.

```
transform: none|transform-functions|initial|inherit;
```

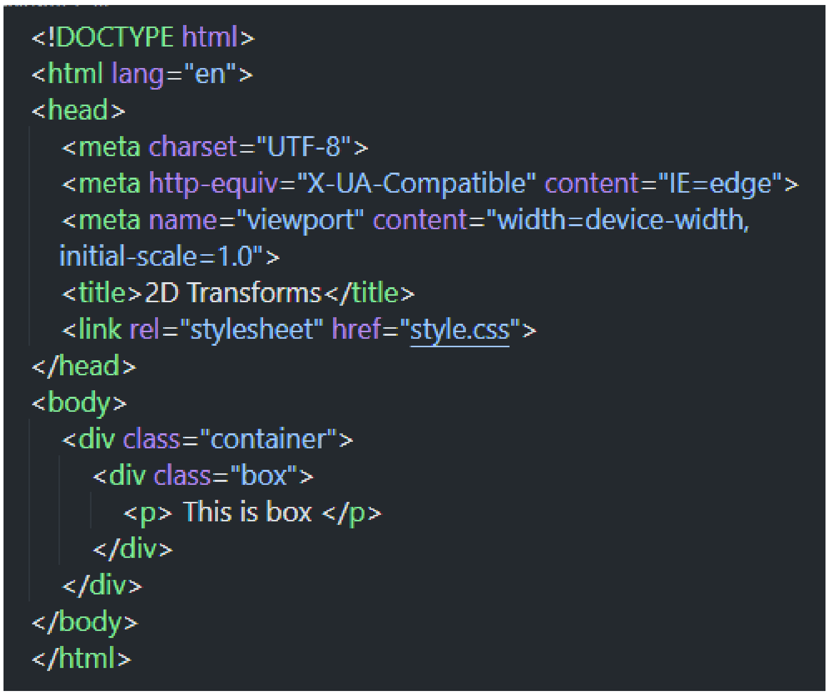
Scale will increase size according to its height and width. It can be scale(2), scaleX(2), scaleY(2), scaleZ(2), scale(2, 2) here first one is x and second is y

Skew(45deg) by this we change its angle. Skew(10deg, 40deg)

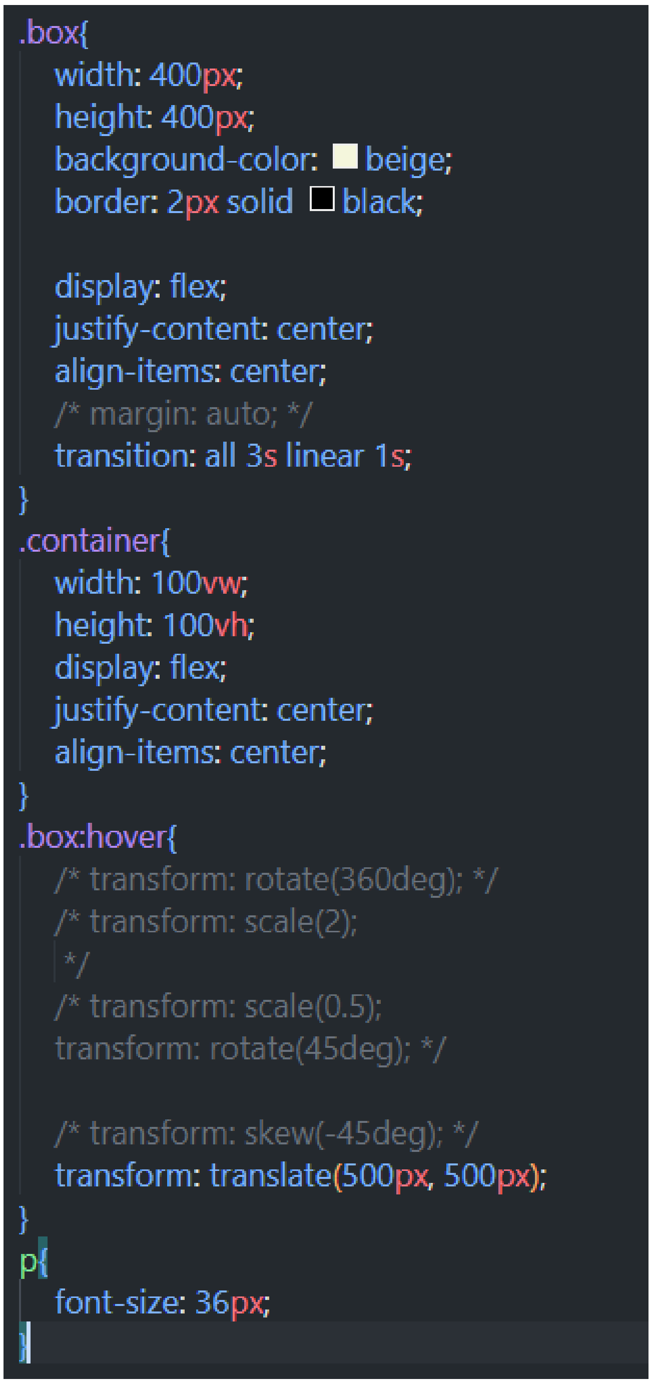
Translate by using this we will move anything from one place to another place with respect to previous location. Translate(20px)   translate(20px , 40px) first is x axis and other is z axis. Translate(50%, 50%)

Rotate(20deg), rotateX(40deg), rotateY(40deg)

Transform: roate(45deg) scale(1.2) skew(30deg) translate(45px)



```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>2D Transforms</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <div class="box">
            <p> This is box </p>
        </div>
    </div>
</body>
</html>
```



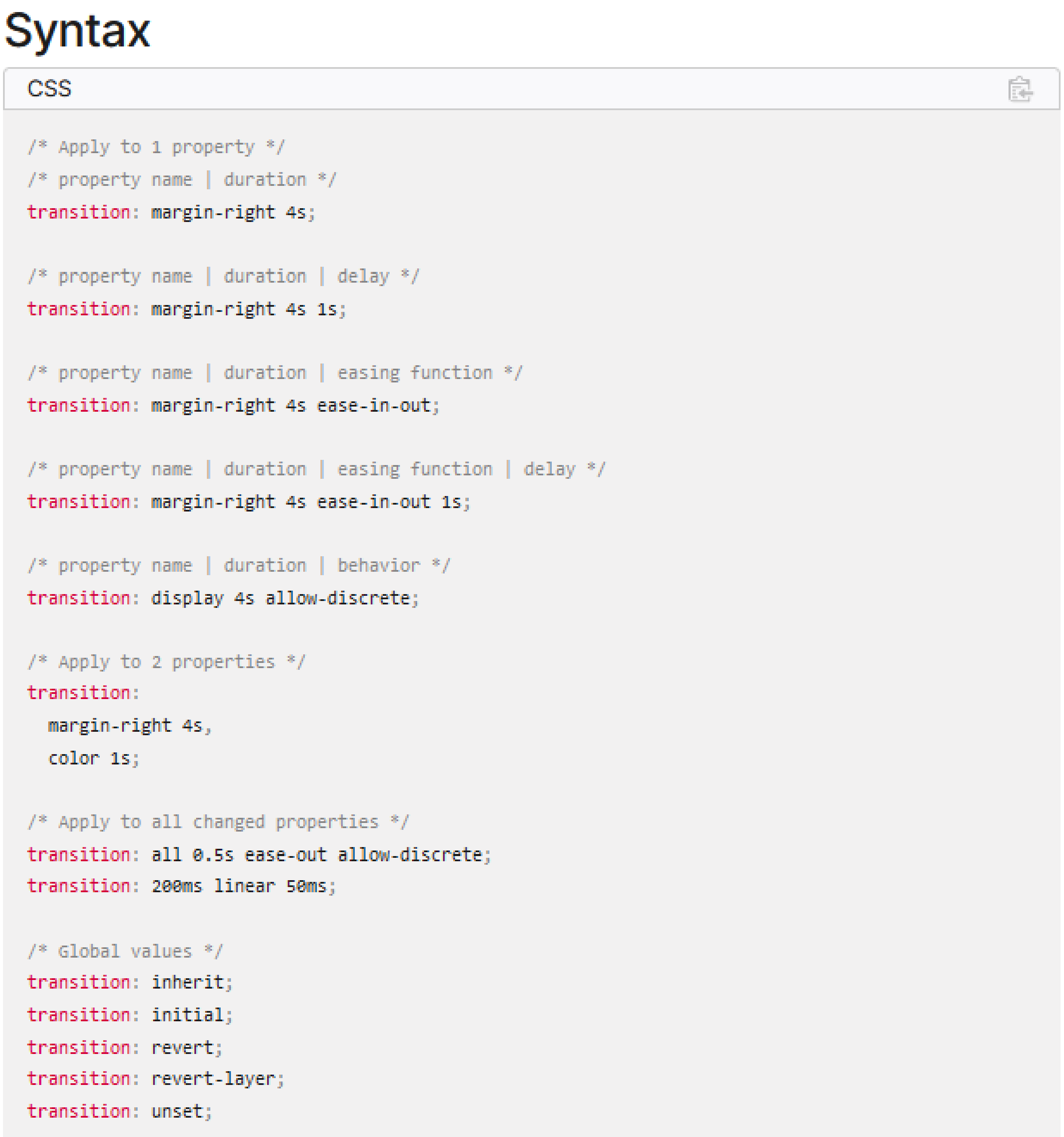
```css
.box{
    width: 400px;
    height: 400px;
    background-color: beige;
    border: 2px solid black;

    display: flex;
    justify-content: center;
    align-items: center;
    /* margin: auto; */
    transition: all 3s linear 1s;
}
.container{
    width: 100vw;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
}
.box:hover{
    /* transform: rotate(360deg); */
    /* transform: scale(2);
    */
    /* transform: scale(0.5);
    transform: rotate(45deg); */

    /* transform: skew(-45deg); */
    transform: translate(500px, 500px);
}
p{
    font-size: 36px;
}
```

Transition in CSS => we can say that animation is parent and transition is child

How to move from one stage of styling to another stage of styling this process is called Transition. Transitions in CSS allow us to control the way in which transition takes place between the two states of the element. For example, when hovering your mouse over a button, you can change the background color of the element with help of CSS selector and pseudo-class. There are four CSS properties that you should use, all or in part (transition-timing-function, transition-property and transition-duration, transition delay), to animate the transition. All these properties must be placed along with other CSS properties of the initial state of the element: we can apply transition between transform . Transition work when we perform any action like hover.

Transition by default property is all. So instead of using al we write property one by one becz if we choose all then in low hand devices website will lags.

Css animation properties in MDN see all properties we can apply transition



Syntax

CSS

```css
/* Apply to 1 property */
/* property name | duration */
transition: margin-right 4s;

/* property name | duration | delay */
transition: margin-right 4s 1s;

/* property name | duration | easing function */
transition: margin-right 4s ease-in-out;

/* property name | duration | easing function | delay */
transition: margin-right 4s ease-in-out 1s;

/* property name | duration | behavior */
transition: display 4s allow-discrete;

/* Apply to 2 properties */
transition:
  margin-right 4s,
  color 1s;

/* Apply to all changed properties */
transition: all 0.5s ease-out allow-discrete;
transition: 200ms linear 50ms;

/* Global values */
transition: inherit;
transition: initial;
transition: revert;
transition: revert-layer;
transition: unset;
```

The `transition-timing-function` property, normally used as part of `transition` shorthand, is used to define a function that describes how a transition will proceed over its duration, allowing a transition to change speed during its course.

```
.example {
    transition-timing-function: ease-out;
}
```

These timing functions are commonly called *easing functions*, and can be defined using a predefined keyword value, a stepping function, or a cubic Bézier curve.

The predefined keyword values allowed are:

- ease
- linear
- ease-in
- ease-out
- ease-in-out
- step-start
- step-end

For some values, the effect may not be as obvious unless the transition duration is set to a larger value.

Each of the predefined keywords has an equivalent cubic Bézier curve value or equivalent stepping function value. For example, the following two timing function values would be equivalent to each other:

```
.example {
    transition-timing-function: ease-out;
}

.example-2 {
    transition-timing-function: cubic-bezier(0, 0, 0.58, 1);
}
```

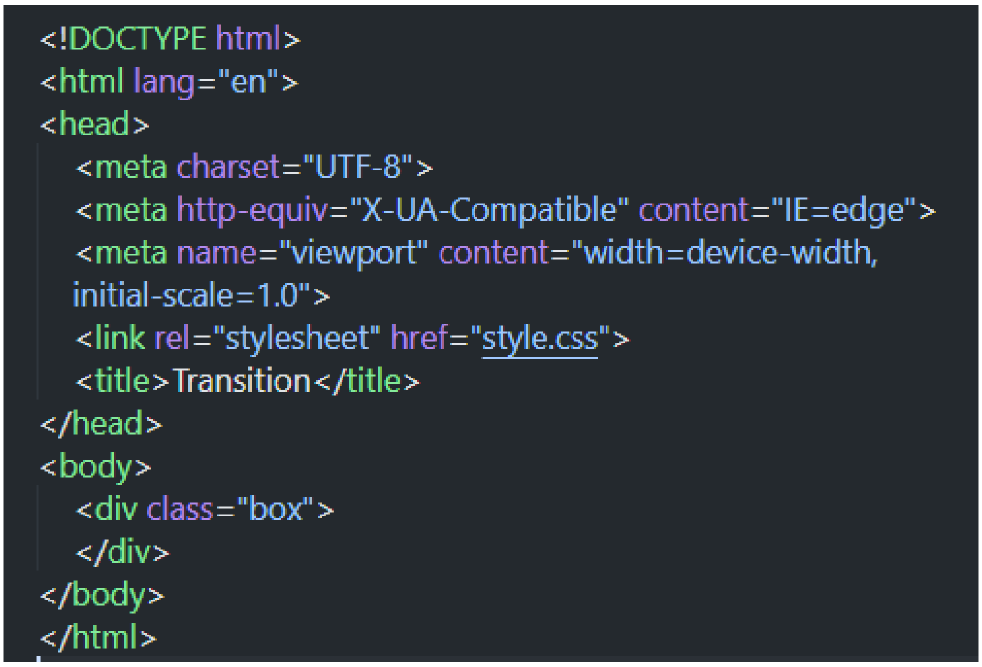
As would the following two:

```
.example {
    transition-timing-function: step-start;
}

.example-2 {
    transition-timing-function: steps(1, start);
}
```

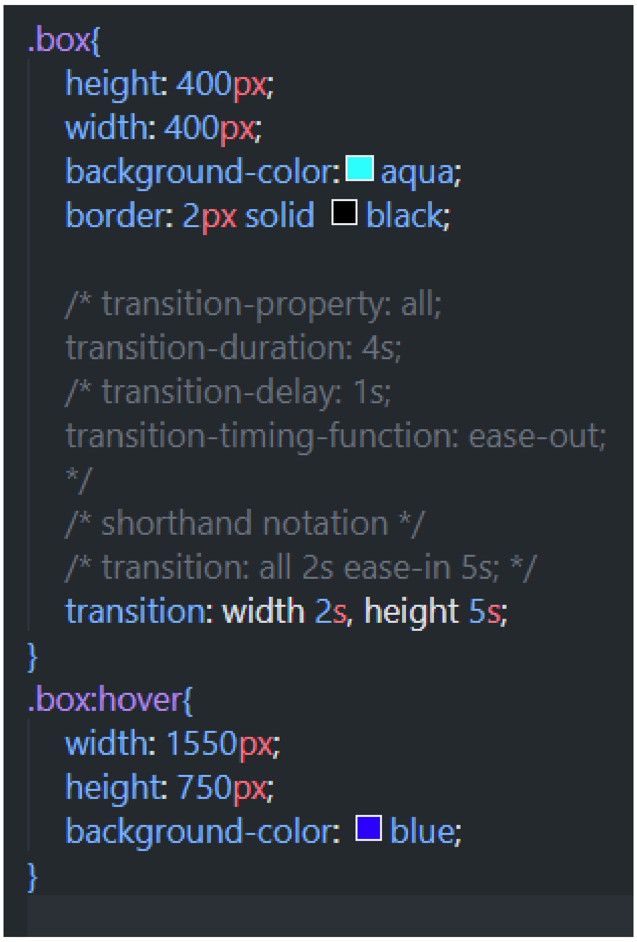
Using steps() and Bézier curves =>

The `steps()` function allows you to specify intervals for the timing function. It takes one or two parameters, separated by a comma: a positive integer and an optional value of either `start` or `end`. If no second parameter is included, it will default to `end`.

To understand stepping functions, here is a demo that uses `steps(4, start)` for the box on the left, and `steps(4, end)` for the box on the right (hover over a box or reload the frame to view the transitions):



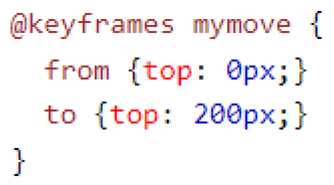
```html
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport" content="width=device-width,
  initial-scale=1.0">
   <link rel="stylesheet" href="style.css">
   <title>Transition</title>
</head>
<body>
   <div class="box">
   </div>
</body>
</html>
```



```css
.box{
   height: 400px;
   width: 400px;
   background-color: aqua;
   border: 2px solid  black;

   /* transition-property: all;
   transition-duration: 4s;
   /* transition-delay: 1s;
   transition-timing-function: ease-out;
   */
   /* shorthand notation */
   /* transition: all 2s ease-in 5s; */
   transition: width 2s, height 5s;
}
.box:hover{
   width: 1550px;
   height: 750px;
   background-color:  blue;
}
```

Animation in CSS=>

When a particular element or component shift form one style to another style this process is called animation. The information of shifting form one style to another style is saved in @keyframes.



```
@keyframes mymove {
  from {top: 0px;}
  to {top: 200px;}
}
```

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

Transition work when we perform any action like hover.

By the help of transition we can only tell from where we have to go to where we have to go but by using key frame we can do any thing

Basic Declaration & Usage

```
@keyframes name-of-animation {
  0%    { opacity: 0; }
  20%   { opacity: 1; }
  80%   { opacity: 0; }
  100%  { opacity: 1; }
}

.animate-this-element {
  animation: name-of-animation 5s infinite;
}
```

You can use any number of “stops” in the @keyframe animation, and it’s one of the main strengths of keyframe animations. While CSS transition is only from one state to another, a keyframe animation can interpolate between many different states during its timeline.

If an animation has the same starting and ending properties, one way to do that is to comma-separate the 0% and 100% values:

```
@keyframes fontbulger {
  0%, 100% {
    font-size: 10px;
  }
  50% {
    font-size: 12px;
  }
}
```

Or, you could always tell the animation to run twice (or any even number of times) and tell the direction to alternate.

Calling keyframe animation with separate properties

```
.box {
 animation-name: bounce;
 animation-duration: 4s; /* or: Xms */
 animation-iteration-count: 10;
 animation-direction: alternate; /* or: normal */
 animation-timing-function: ease-out; /* or: ease, ease-in, ease-in-out, linear, cubic-
bezier(x1, y1, x2, y2) */
 animation-fill-mode: forwards; /* or: backwards, both, none */
 animation-delay: 2s; /* or: Xms */
}
```

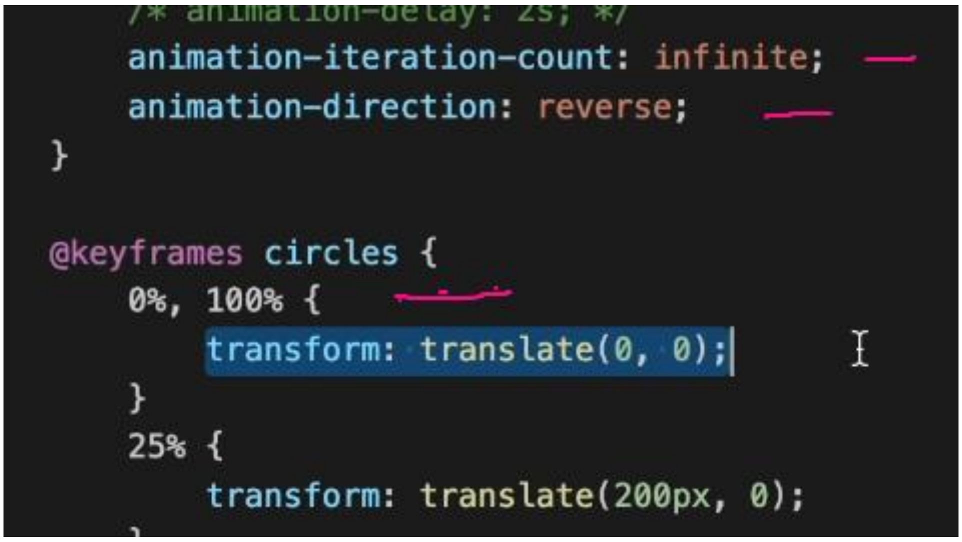
Animation Shorthand

Just space-separate all the individual values. The order doesn’t matter except when using both duration and delay, they need to be in that order. In the example below 1s = duration, 2s = delay, 3 = iterations.

```
animation: test 1s 2s 3 alternate backwards;
```

Animation direction =>

Normal, reverse, alternate, alternate reverse



```
    /* animation-delay: 2s; */
    animation-iteration-count: infinite;
    animation-direction: reverse;
}

@keyframes circles {
    0%, 100% {
        transform: translate(0, 0);
    }
    25% {
        transform: translate(200px, 0);
```