CSS3

CSS Selectors

- A CSS selector selects the HTML element(s) you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

The CSS element Selector

The element selector selects HTML elements based on the element name.

Example

Here, all <p> elements on the page will be center-aligned, with a red text color

```
p {

  text-align: center;

  color: red;

}
```

The CSS id Selector

- The id selector uses the id attribute of an HTML element to select a specific element.
- The id of an element is unique within a page, so the id selector is used to select one unique element!
- To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example:

The CSS rule below will be applied to the HTML element with id="para1":

```css
#para1 {

  text-align: center;

  color: red;

}
```

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

Example

In this example all HTML elements with class="center" will be red and center-aligned:

```css
.center {

  text-align: center;

  color: red;

}
```

The CSS Universal Selector

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

Example

The CSS rule below will affect every HTML element 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.

Example CSS code (the h1, h2, and p elements have the same style definitions):

```
h1, h2, p {

  text-align: center;

  color: red;

}
```

How To Add CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

Example

External styles are defined within the <link> element, inside the <head> section of an HTML page

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

```html
<!DOCTYPE html>

<html>

<head>

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

</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>
```

"mystyle.css"

```
body {

  background-color: lightblue;

}


h1 {

  color: navy;

  margin-left: 20px;

}
```

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

```
<!DOCTYPE html>

<html>

<head>

<style>

body {

  background-color: linen;}

h1 {

  color: maroon;

  margin-left: 40px;}

</style>
```

```
</head>

<body>

<h1>This is a heading</h1>

<p>This is a paragraph.</p>

</body>

</html>
```

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Example

```
<!DOCTYPE html>

<html>

<body>

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

<p style="color:red;">This is a paragraph.</p>

</body>

</html>
```

Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Assume that an external style sheet has the following style for the <h1> element:

```
h1 {

  color: navy; }
```

Then, assume that an internal style sheet also has the following style for the <h1> element:

```
h1 {

  color: orange;     }
```

If the internal style is defined after the link to the external style sheet, the <h1> elements will be "orange":

However, if the internal style is defined before the link to the external style sheet, the <h1> elements will be "navy":

Cascading Order

What style will be used when there is more than one style specified for an HTML element?

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

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

So, an inline style has the highest priority, and will override external and internal styles and browser defaults.

CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

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

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

p {

  color: red;

}
```

CSS Borders

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.

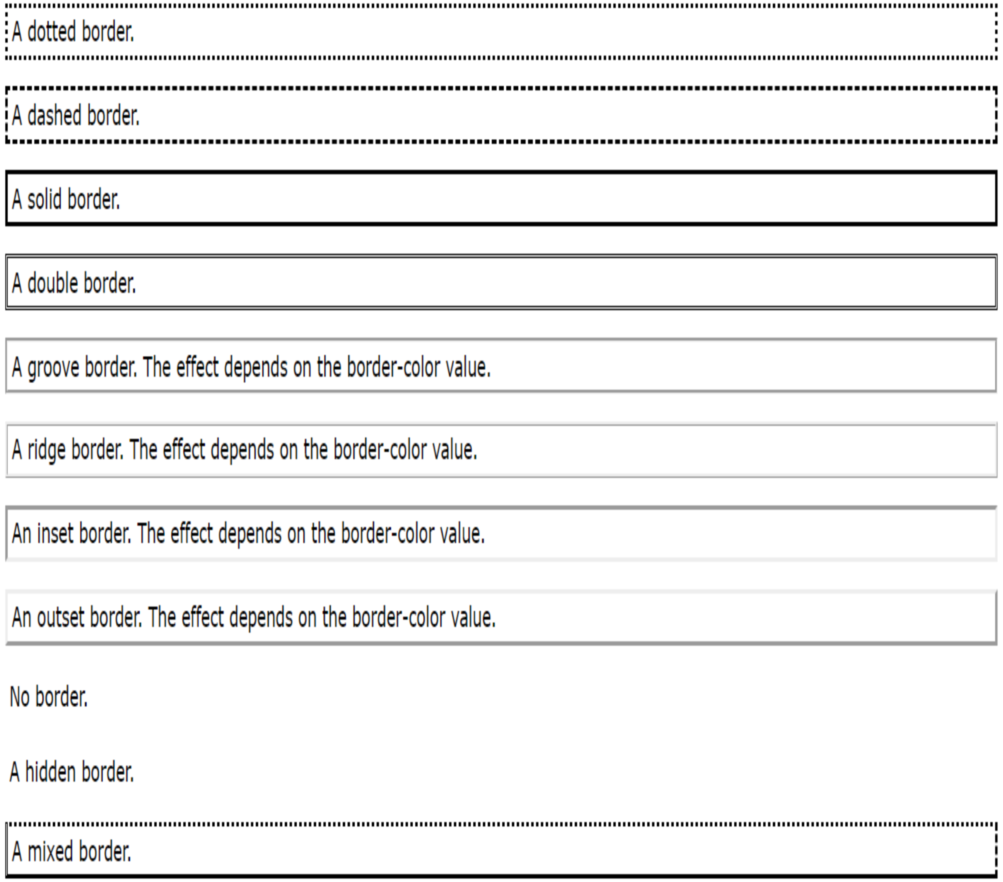
The `border-style` property can have from one to four values (for the top border, right border, bottom border, and the left border).

CSS Border Style

- `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

Example

```
p.dotted {border-style: dotted;}

p.dashed {border-style: dashed;}

p.solid {border-style: solid;}

p.double {border-style: double;}

p.groove {border-style: groove;}

p.ridge {border-style: ridge;}

p.inset {border-style: inset;}

p.outset {border-style: outset;}

p.none {border-style: none;}

p.hidden {border-style: hidden;}

p.mix {border-style: dotted dashed solid double;}
```



A dotted border.

A dashed border.

A solid border.

A double border.

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

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

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

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

No border.

A hidden border.

A mixed border.

CSS Border Width

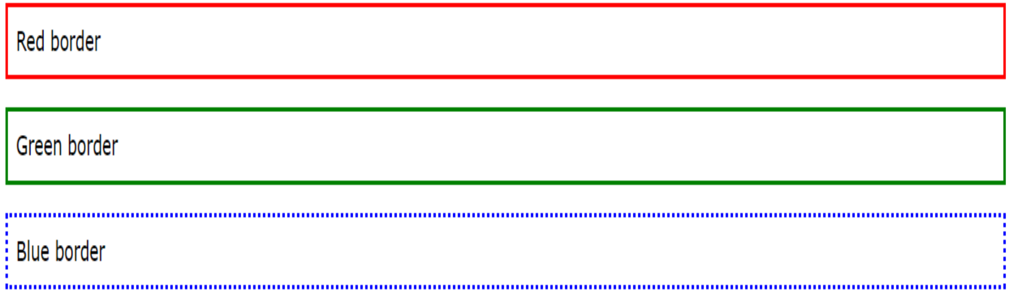
The `border-width` property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick.

Specific Side Widths

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

CSS Border Color



The `border-color` property is used to set the color of the four borders.

The color can be set by:

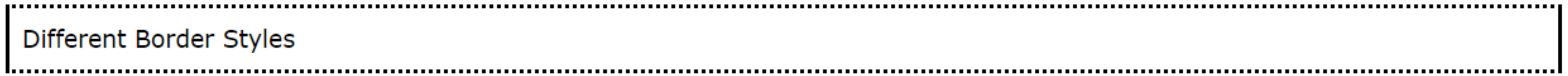
- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

If `border-color` is not set, it inherits the color of the element.

CSS Border - Individual Sides

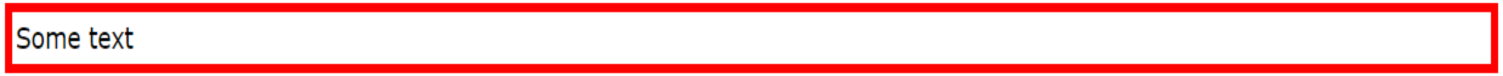
In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left).

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



Different Border Styles

CSS Border - Shorthand Property



To shorten the code, it is also possible to specify all the individual border properties in one property.

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

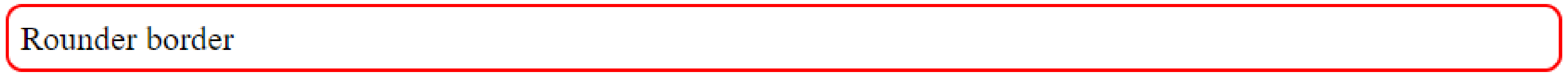
- `border-width`
- `border-style` (required)
- `border-color`

```
p {

  border: 5px solid red;

}
```

CSS Rounded Borders

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

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



Rounder border

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.

The property has three parts:

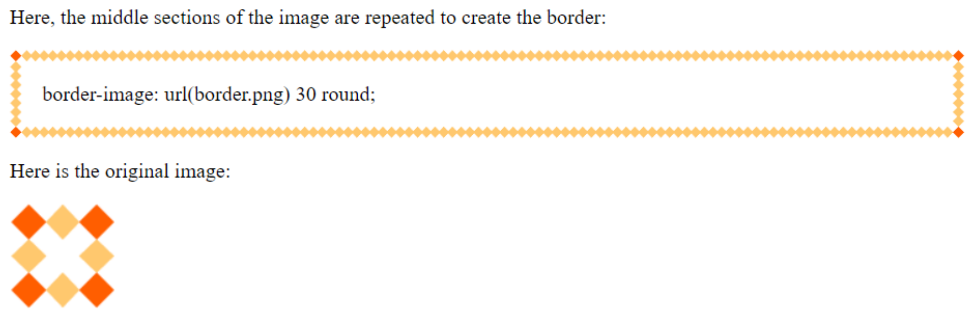
1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

CSS border-image Property

The `border-image` property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

For `border-image` to work, the element also needs the `border` property set!

```
<html>
<head>
<style>
#borderimg {
  border: 10px solid transparent;
  padding: 15px;
  border-image: url(border.png) 30 round;
}
</style>
</head>
<body>

<h1>The border-image Property</h1>

<p>Here, the middle sections of the image are repeated to create the border:</p>
<p id="borderimg">border-image: url(border.png) 30 round;</p>
```


Here, the middle sections of the image are repeated to create the border:

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

Here is the original image:

This element has a margin of 70px.

CSS Margins

Margins are used to create space around elements, outside of any defined borders.

Margin - Individual Sides

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

CSS Padding

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

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

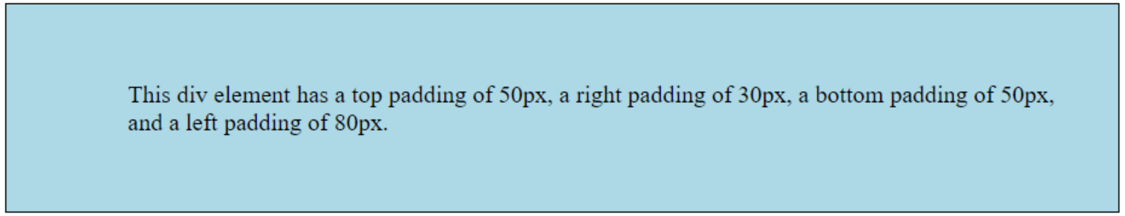
- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

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

CSS Padding Example

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```


This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px, and a left padding of 80px.

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.

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.

```
#div1 {
  background: url(img_flower.jpg);
  background-size: 100px 80px;
  background-repeat: no-repeat;
}
```

CSS Background Size

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

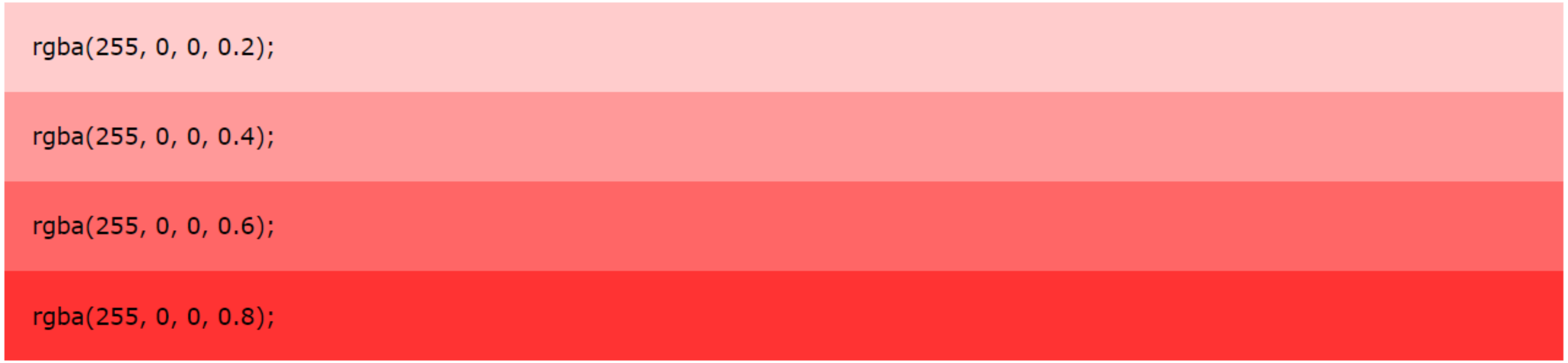
The `contain` keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.

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). As such, some parts of the background image may not be visible in the background positioning area.

CSS Colors

CSS supports 140+ color names, HEX values, RGB values, RGBA values, HSL values, HSLA values, and opacity.

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).



```
rgba(255, 0, 0, 0.2);
rgba(255, 0, 0, 0.4);
rgba(255, 0, 0, 0.6);
rgba(255, 0, 0, 0.8);
```

CSS Shadow Effects

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

- `text-shadow`
- `box-shadow`

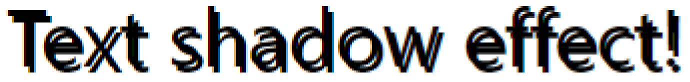
CSS Text Shadow

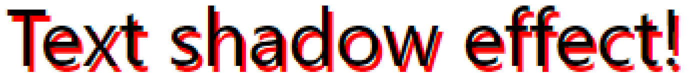
The CSS `text-shadow` property applies shadow to text.

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

Example

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

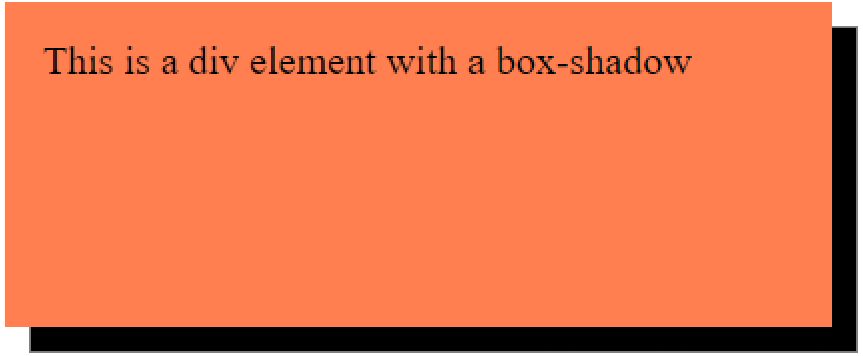

Text shadow effect!


Text shadow effect!

CSS box-shadow Property

The CSS `box-shadow` property applies shadow to elements.

```
<style>
div {
  width: 300px;
  height: 100px;
  padding: 15px;
  background-color: coral;
  box-shadow: 10px 10px;
}
</style>
```



CSS Text Effects

- `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.

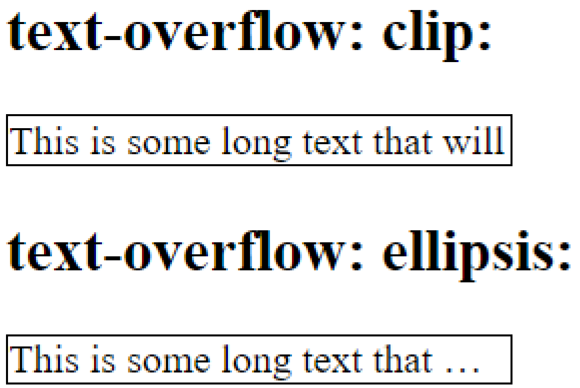
It can be clipped:

This is some long text that will not fit in the box

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

This is some long text that w

```
text-overflow: clip;

text-overflow: ellipsis;
```



text-overflow: clip:

This is some long text that will

text-overflow: ellipsis:

This is some long text that …

CSS Word Wrapping

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

Allow long words to be able to be broken and wrap onto the next line:

```css
p {
  word-wrap: break-word;
}
```

CSS Word Breaking

The CSS `word-break` property specifies line breaking rules.

```
p.test1 {

  word-break: keep-all;

}


p.test2 {

  word-break: break-all;

}
```

CSS Writing Mode

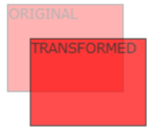
The CSS `writing-mode` property specifies whether lines of text are laid out horizontally or vertically.

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

```
p.test1 {

  writing-mode: horizontal-tb;

}

span.test2 {

  writing-mode: vertical-rl;
```

CSS 2D Transforms

The translate() Method



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

- `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).

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

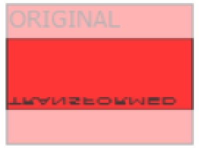
CSS 3D Transforms

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

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

```
#myDiv {

  transform: rotateX(150deg);

}
```

The rotateX() Method



CSS Transitions

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

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

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

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

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

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

CSS Animations

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

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 `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.

Delay an Animation

The `animation-delay` property specifies a delay for the start of an animation.

Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for *N* seconds.

Set How Many Times an Animation Should Run

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

Run Animation in Reverse Direction or Alternate Cycles

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

Specify the Speed Curve of the Animation

The `animation-timing-function` property specifies the speed curve of the animation.

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

- `ease` - Specifies an animation with a slow start, then fast, then end slowly (this is default)
- `linear` - Specifies an animation with the same speed from start to end
- `ease-in` - Specifies an animation with a slow start
- `ease-out` - Specifies an animation with a slow end
- `ease-in-out` - Specifies an animation with a slow start and end
- `cubic-bezier(n,n,n,n)` - Lets you define your own values in a cubic-bezier function

Specify the fill-mode For an Animation

CSS animations do not affect an element before the first keyframe is played or after the last keyframe is played. The animation-fill-mode property can override this behavior.

The `animation-fill-mode` property specifies a style for the target element when the animation is not playing (before it starts, after it ends, or both).

The animation-fill-mode property can have the following values:

- `none` - Default value. Animation will not apply any styles to the element before or after it is executing
- `forwards` - The element will retain the style values that is set by the last keyframe (depends on animation-direction and animation-iteration-count)
- `backwards` - The element will get the style values that is set by the first keyframe (depends on animation-direction), and retain this during the animation-delay period
- `both` - The animation will follow the rules for both forwards and backwards, extending the animation properties in both directions

Animation Shorthand Property

```
div {

  animation-name: example;

  animation-duration: 5s;

  animation-timing-function: linear;

  animation-delay: 2s;

  animation-iteration-count: infinite;

  animation-direction: alternate;

} It can be replaced by

div {

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