

Font Awesome Icons

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

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
```

```
<i class="fa fa-cloud"></i>
```

Bootstrap Icons

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

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

Note: No downloading or installation is required!

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

Google Icons

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

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

Note: No downloading or installation is required!

```
<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>
```

CSS Overflow

The CSS `overflow` property specifies whether to clip content or to add scrollbars **when the content of an element is too big to fit in a specified area.**

The `overflow` property has the following values:

- **visible - Default.** The overflow is not clipped. It 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, but a scrollbar is added to see the rest of the content
- **auto** - If overflow is clipped, a scrollbar should be added to see the rest of the content

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

Note: In OS X Lion (on Mac), scrollbars are hidden by default and only shown when being used (even though "overflow:scroll" is set).

CSS Combinators

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS3:

- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

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

The following example selects all <p> elements inside <div> elements:

Example

```
div p {  
    background-color: yellow;  
}
```

Child Selector

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

The following example selects all <p> elements that are immediate children of a <div> element:

Example

```
div > p {  
    background-color: yellow;  
}
```

Adjacent Sibling Selector

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

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

The following example selects all <p> elements that are placed **immediately after** <div> elements:

Example

```
div + p {  
    background-color: yellow;  
}
```

General Sibling Selector

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

The following example selects all <p> elements that are siblings of <div> elements:

Example

```
div ~ p {  
    background-color: yellow;  
}
```

CSS Opacity / Transparency

Transparent Image

The `opacity` property can take a value from 0.0 (.1, .2, .3, ..., .9) – 1 (as it is). The lower value, the more transparent:

Note: IE8 and earlier use `filter: alpha(opacity=x)`. The x can take a value from 0 - 100. A lower value makes the element more transparent.

```
img {  
    opacity: 0.5;  
    filter: alpha(opacity=50); /* For IE8 and earlier */  
}
```

Transparent Hover Effect

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

```
img {  
    opacity: 0.5;  
    filter: alpha(opacity=50); /* For IE8 and earlier */  
}  
  
img:hover {  
    opacity: 1.0;  
    filter: alpha(opacity=100); /* For IE8 and earlier */  
}
```

Transparent Box

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

Example

```
div {  
    opacity: 0.3;  
    filter: alpha(opacity=30); /* For IE8 and earlier */  
}
```

Transparency using RGBA

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

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).

CSS3 Rounded Corners

With the CSS3 `border-radius` property, you can give any element "rounded corners".

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

CSS3 border-radius - Specify Each Corner

If you specify only one value for the `border-radius` property, this radius will be applied to all 4 corners.

However, you can specify each corner separately if you wish. Here are the rules:

- **Four values:** first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner
- **Three values:** first value applies to top-left, second value applies to top-right and bottom-left, and third value applies to bottom-right
- **Two values:** first value applies to top-left and bottom-right corner, and the second value applies to top-right and bottom-left corner
- **One value:** all four corners are rounded equally

CSS3 background-origin Property

The CSS3 `background-origin` property specifies where the background image is positioned.

The property takes three different values:

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

The following example illustrates the `background-origin` property:

Example

```
#example1 {  
  border: 10px solid black;  
  padding: 35px;  
  background: url(img_flwr.gif);  
  background-repeat: no-repeat;  
  background-origin: content-box;  
}
```

CSS3 background-clip Property

The CSS3 `background-clip` property specifies the painting area of the background.

The property takes three different values:

- `border-box` - **(default)** the background is painted to the outside edge of the border
- `padding-box` - the background is painted to the outside edge of the padding

- content-box - the background is painted within the content box

The following example illustrates the `background-clip` property:

Example

```
#example1 {  
  border: 10px dotted black;  
  padding: 35px;  
  background: yellow;  
  background-clip: content-box;  
}
```

CSS3 Colors

CSS supports color names, hexadecimal and RGB colors.

In addition, CSS3 also introduces:

- RGB colors
- RGBA colors
- HSL colors
- HSLA colors
- opacity

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).

```
rgba(255, 0, 0, 0.2);
```

HSL Colors

HSL stands for Hue, Saturation and Lightness.

An HSL color value is specified with: `hsl(hue, saturation, lightness)`.

1. Hue is a degree on the color wheel (from 0 to 360):
 - 0 (or 360) is red
 - 120 is green

- 240 is blue
- 2. Saturation is a percentage value: 100% is the full color.
- 3. Lightness is also a percentage; 0% is dark (black) and 100% is white.

```
hsl(0, 100%, 30%);  
hsl(0, 100%, 50%);
```

HSLA Colors

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

An HSLA color value is specified with: `hsla(hue, saturation, lightness, alpha)`, where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
hsla(0, 100%, 30%, 0.3);
```

Opacity

The CSS3 `opacity` property sets the opacity for the whole element (both background color and text will be opaque/transparent).

The `opacity` property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

```
rgb(255, 0, 0);opacity:0.2;  
rgb(255, 0, 0);opacity:0.4;
```

CSS3 Shadow Effects

With CSS3 you can add shadow to text and to elements.

In this chapter you will learn about the following properties:

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

CSS3 Text Shadow

The CSS3 `text-shadow` property applies shadow to text.

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

Text shadow effect!

Example

```
h1 {  
  text-shadow: 2px 2px;  
}
```

Next, add a color to the shadow:

Text shadow effect!

Example

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

hen, add a blur effect to the shadow:

Text shadow effect!

Example

```
h1 {  
  text-shadow: 2px 2px 5px red;  
}
```

The following example shows a white text with black shadow:

Text shadow effect!

Example

```
h1 {  
  color: white;  
  text-shadow: 2px 2px 4px #000000;  
}
```

CSS3 Text

CSS3 contains several new text features.

In this chapter you will learn about the following text properties:

- text-overflow
- word-wrap
- word-break

CSS3 Text Overflow

The CSS3 `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 will not fit in the box

The CSS code is as follows:

Example

```
p.test1 {  
  white-space: nowrap;  
  width: 200px;  
  border: 1px solid #000000;  
  overflow: hidden;  
  text-overflow: clip;  
}
```

CSS3 Web Fonts

CSS3 Web Fonts - The @font-face Rule






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

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

Your "own" fonts are defined within the CSS3 `@font-face` rule.

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

Property					
@font-face	4.0	9.0	3.5	3.2	10.0

Different Font Formats

TrueType Fonts (TTF)

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

OpenType Fonts (OTF)

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

The Web Open Font Format (WOFF)

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

The Web Open Font Format (WOFF 2.0)

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

SVG Fonts/Shapes






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

Embedded OpenType Fonts (EOT)

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

Browser Support for Font Formats

The numbers in the table specifies the first browser version that fully supports the font format.

Font format					
TTF/OTF	9.0*	4.0	3.5	3.1	10.0
WOFF	9.0	5.0	3.6	5.1	11.1
WOFF2	Not supported	36.0	35.0*	Not supported	26.0
SVG	Not supported	4.0	Not supported	3.2	9.0
EOT	6.0	Not supported	Not supported	Not supported	Not supported

*IE: The font format only works when set to be "installable".

*Firefox: Not supported by default, but can be enabled (need to set a flag to "true" to use WOFF2).

Using The Font You Want

In the CSS3 `@font-face` rule you must first define a name for the font (e.g. `myFirstFont`), and then point to the font file.

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

To use the font for an HTML element, refer to the name of the font (`myFirstFont`) through the `font-family` property:

Example

```
@font-face {  
  font-family: myFirstFont;  
  src: url(sansation_light.woff);  
}
```

```
div {  
  font-family: myFirstFont;  
}
```

CSS3 Animations

CSS3 animations allows animation of most HTML elements without using JavaScript or Flash!

Browser Support for Animations

The numbers in the table specify the first browser version that fully supports the property.

Numbers followed by -webkit-, -moz-, or -o- specify the first version that worked with a prefix.

What are CSS3 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 you want.

To use CSS3 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;
}
```

Note: If the `animation-duration` property is not specified, the animation will have no effect, because the default value is 0.

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

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

The following example will change the background-color of the `<div>` element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

Example

```
/* The animation code */
@keyframes example {
  0% {background-color: red;}
  25% {background-color: yellow;}
  50% {background-color: blue;}
  100% {background-color: green;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Delay an Animation

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

The following example has a 2 seconds delay before starting the animation:

Example

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

```
    animation-delay: 2s;
}
```

Set How Many Times an Animation Should Run

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

The following example will run the animation 3 times before it stops:

Example

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: 3;
}
```

The following example uses the value "infinite" to make the animation continue for ever:

Example

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
    animation-name: example;
    animation-duration: 4s;
    animation-iteration-count: infinite;
}
```

Run Animation in Reverse Direction or Alternate Cycles

The `animation-direction` property is used to let an animation run in reverse direction or alternate cycles.

The following example will run the animation in reverse direction:

Example

```
div {
    width: 100px;
    height: 100px;
    position: relative;
    background-color: red;
```

```
animation-name: example;  
animation-duration: 4s;  
animation-iteration-count: 3;  
animation-direction: reverse;  
}
```

The following example uses the value "alternate" to make the animation first run forward, then backward, then forward:

Example

```
div {  
  width: 100px;  
  height: 100px;  
  position: relative;  
  background-color: red;  
  animation-name: example;  
  animation-duration: 4s;  
  animation-iteration-count: 3;  
  animation-direction: alternate;  
}
```

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

The following example shows the some of the different speed curves that can be used:

Example

```
#div1 {animation-timing-function: linear;}  
#div2 {animation-timing-function: ease;}  
#div3 {animation-timing-function: ease-in;}  
#div4 {animation-timing-function: ease-out;}  
#div5 {animation-timing-function: ease-in-out;}
```

Animation Shorthand Property

The example below uses six of the animation properties:

Example

```
div {  
  animation-name: example;  
  animation-duration: 5s;  
  animation-timing-function: linear;  
  animation-delay: 2s;  
  animation-iteration-count: infinite;  
  animation-direction: alternate;  
}
```

The same animation effect as above can be achieved by using the shorthand `animation` property:

Example

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

CSS3 Animation Properties

The following table lists the `@keyframes` rule and all the animation properties:

Property	Description
@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties
animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should play in reverse direction or alternate cycles
animation-duration	Specifies how many seconds or milliseconds an animation takes to complete one cycle
animation-fill-mode	Specifies a style for the element when the animation is not playing (when it is finished, or when it has a delay)
animation-iteration-count	Specifies the number of times an animation should be played
animation-name	Specifies the name of the <code>@keyframes</code> animation
animation-play-state	Specifies whether the animation is running or paused
animation-timing-function	Specifies the speed curve of the animation

CSS Buttons

Basic Button Styling

```
.button {  
  background-color: #4CAF50; /* Green */  
  border: none;  
  color: white;  
  padding: 15px 32px;  
  text-align: center;  
  text-decoration: none;  
  display: inline-block;  
  font-size: 16px;  
}
```

Button Colors

```
.button1 {background-color: #4CAF50;} /* Green */  
.button2 {background-color: #008CBA;} /* Blue */  
.button3 {background-color: #f44336;} /* Red */  
.button4 {background-color: #e7e7e7; color: black;} /* Gray */  
.button5 {background-color: #555555;} /* Black */
```

Button Sizes

Use the `font-size` property to change the font size of a button:

Example

```
.button1 {font-size: 10px;}  
.button2 {font-size: 12px;}  
.button3 {font-size: 16px;}  
.button4 {font-size: 20px;}  
.button5 {font-size: 24px;}
```

Use the `padding` property to change the padding of a button:

```
.button1 {padding: 10px 24px;}  
.button2 {padding: 12px 28px;}  
.button3 {padding: 14px 40px;}  
.button4 {padding: 32px 16px;}  
.button5 {padding: 16px;}
```

Use the `border-radius` property to add rounded corners to a button:

Example

```
.button1 {border-radius: 2px;}  
.button2 {border-radius: 4px;}
```

```
.button3 {border-radius: 8px;}
.button4 {border-radius: 12px;}
.button5 {border-radius: 50%;}
```

Colored Button Borders

```
.button1 {
  background-color: white;
  color: black;
  border: 2px solid #4CAF50; /* Green */
}
```

Hoverable Buttons

Use the `:hover` selector to change the style of a button when you move the mouse over it.

Tip: Use the `transition-duration` property to determine the speed of the "hover" effect:

Example

```
.button {
  -webkit-transition-duration: 0.4s; /* Safari */
  transition-duration: 0.4s;
}

.button:hover {
  background-color: #4CAF50; /* Green */
  color: white;
}
```

Shadow Buttons

Use the `box-shadow` property to add shadows to a button:

Example

```
.button1 {
  box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);
}

.button2:hover {
  box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);
}
```

Disabled Buttons

Use the `opacity` property to add transparency to a button (creates a "disabled" look).

Tip: You can also add the `cursor` property with a value of "not-allowed", which will display a "no parking sign" when you mouse over the button:

Example

```
.disabled {  
  opacity: 0.6;  
  cursor: not-allowed;  
}
```